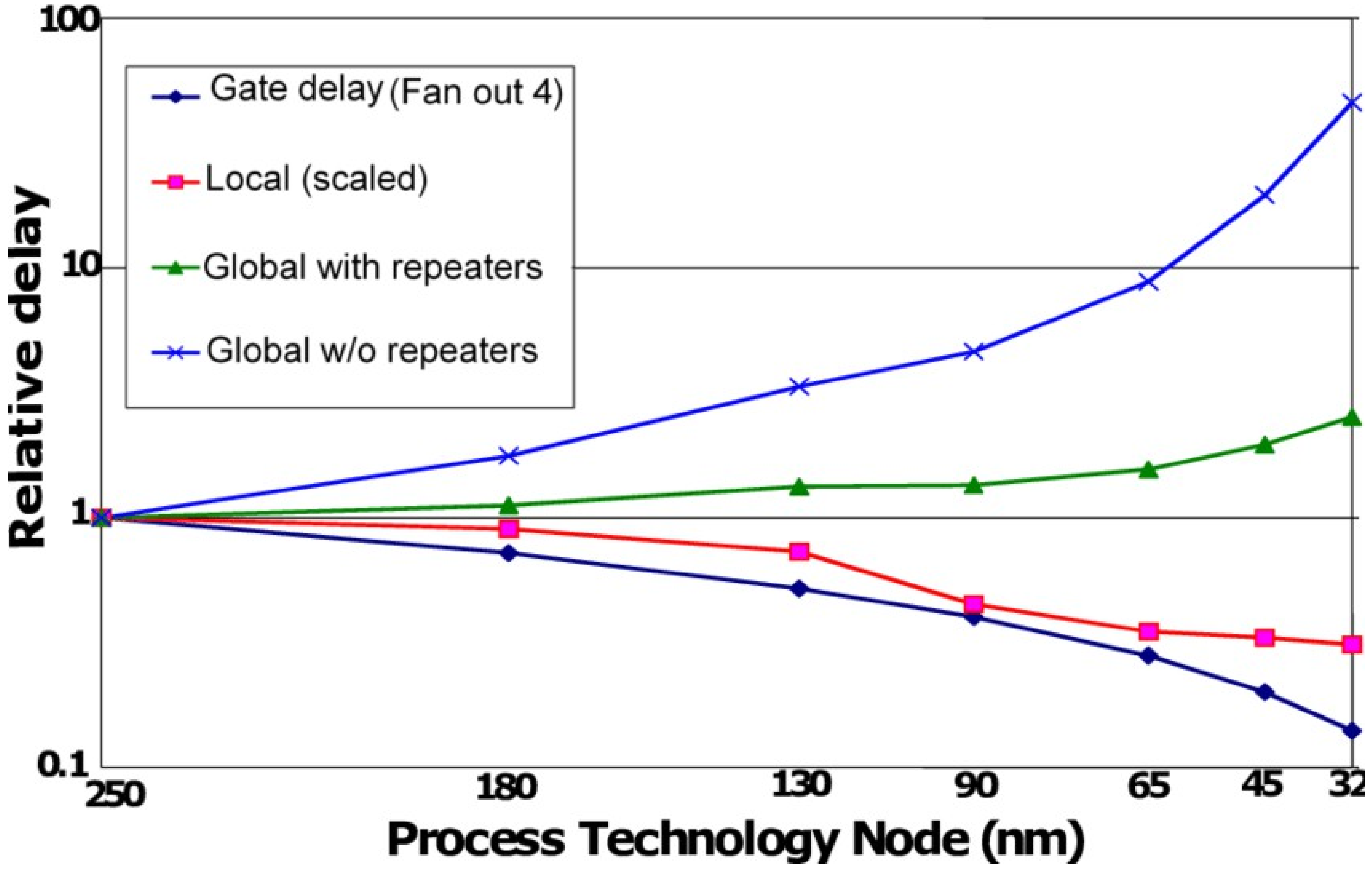
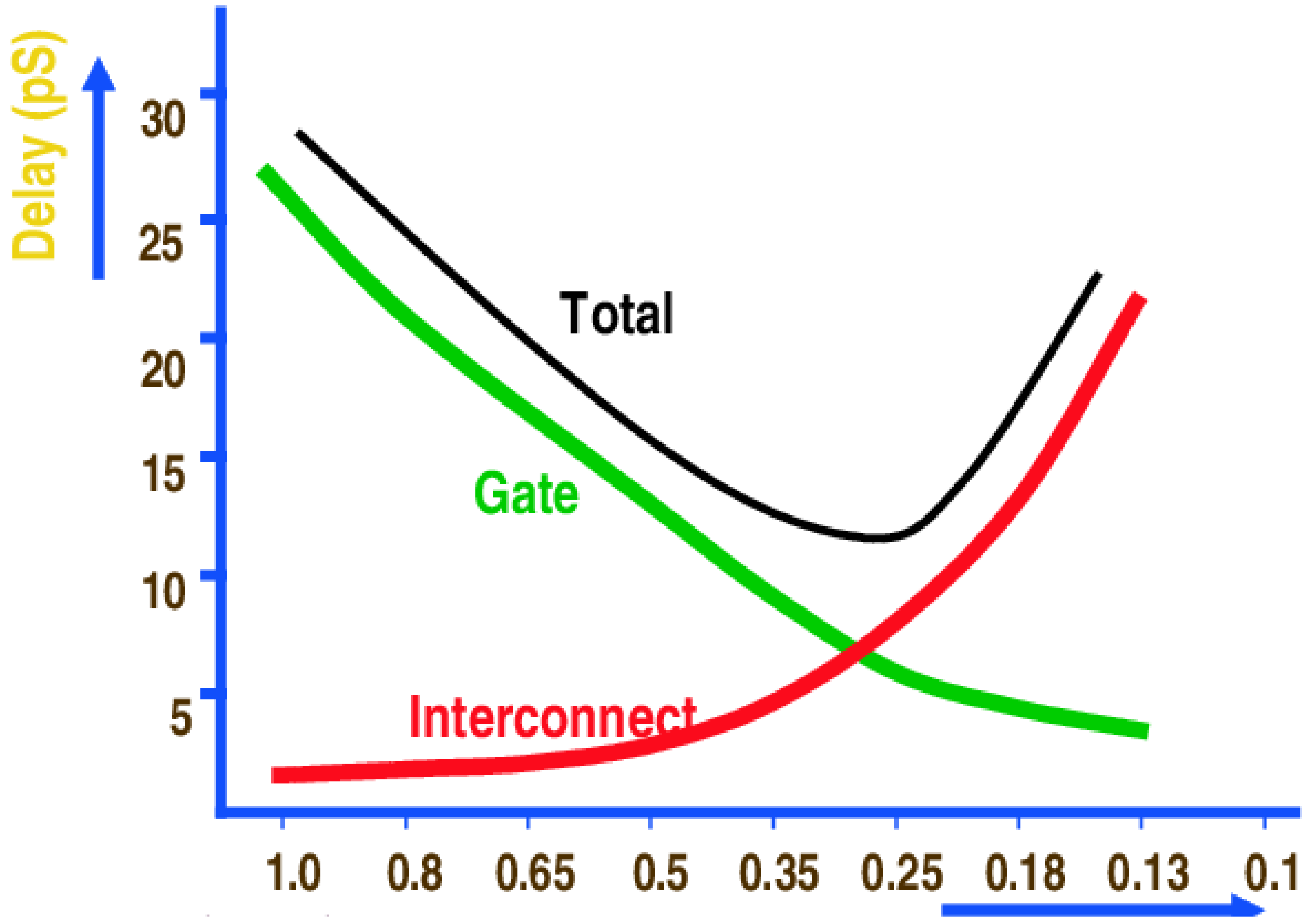
Network on Chip Architectures

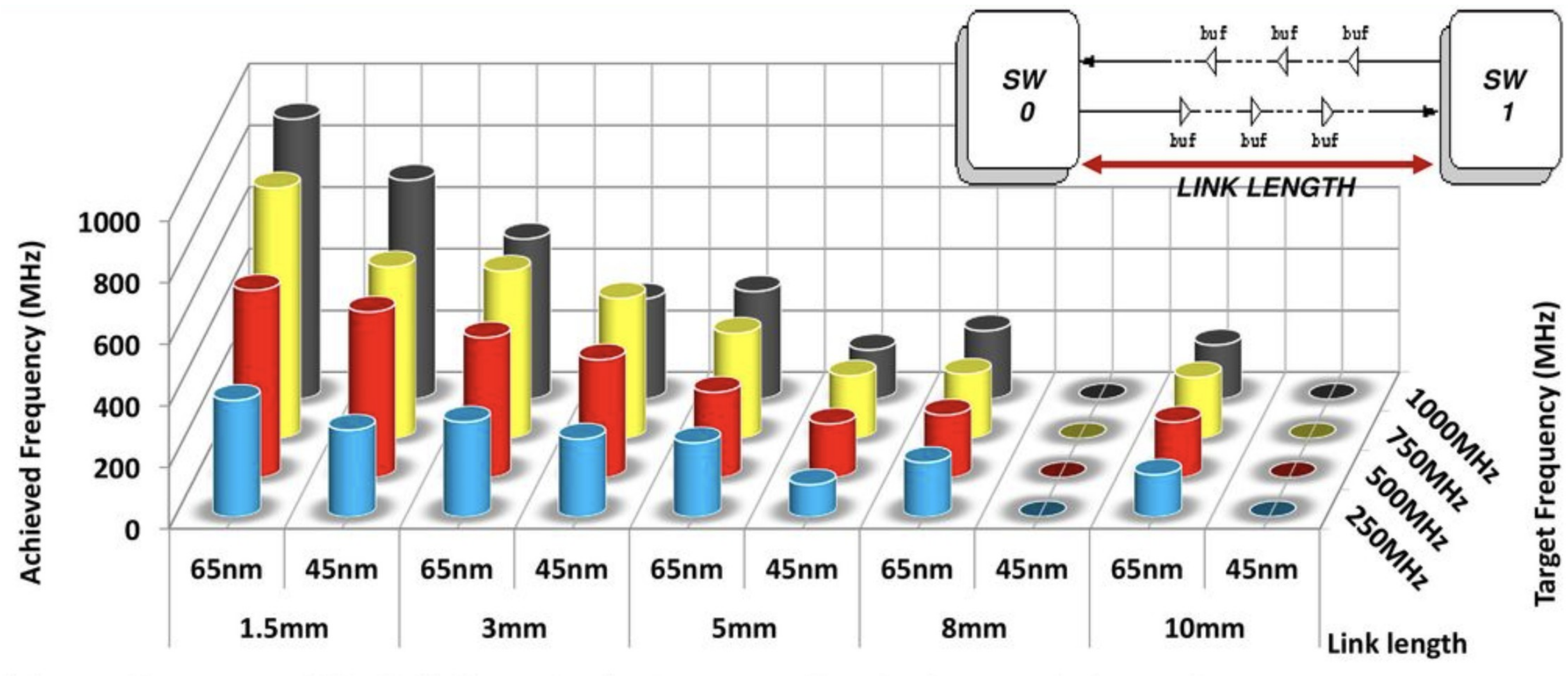
Interconnect Delay Bottleneck



Interconnect Delay Bottleneck

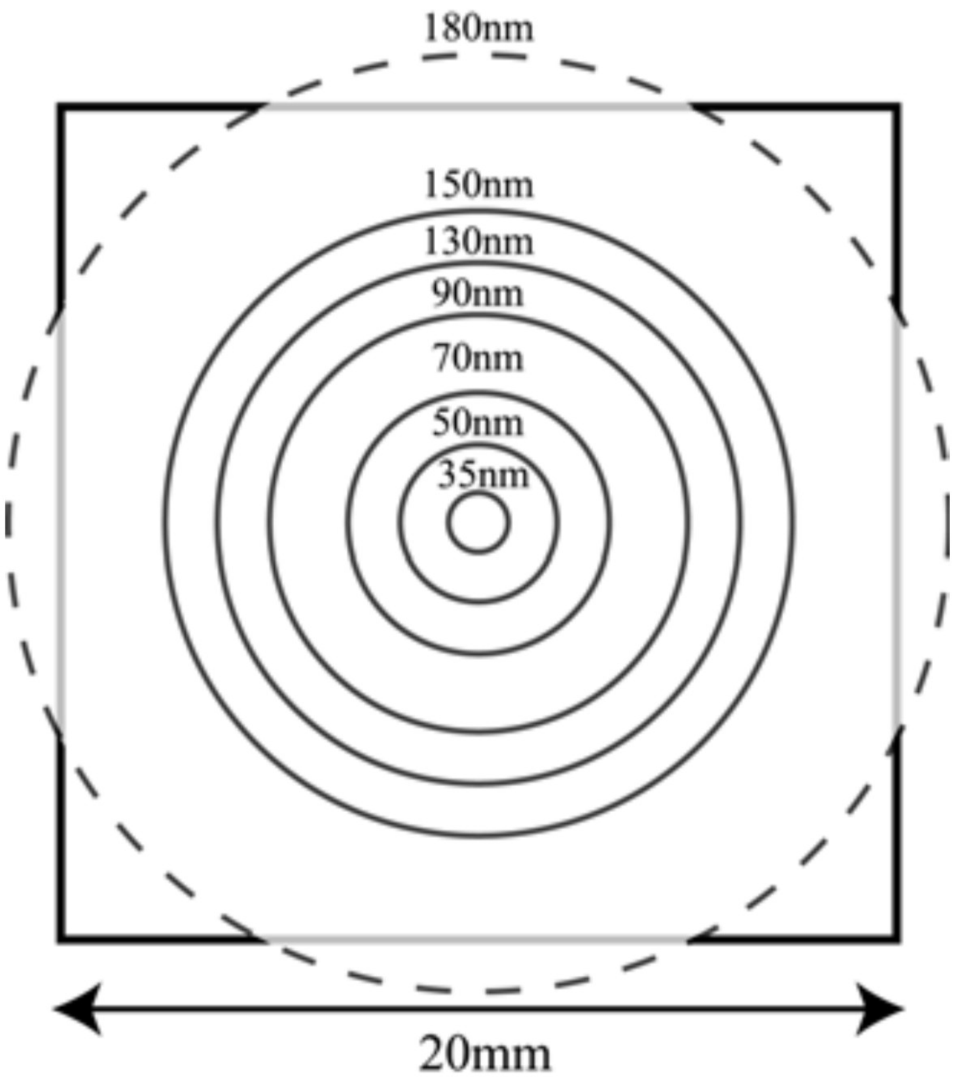


Link Performance



Interconnect Delay Bottleneck

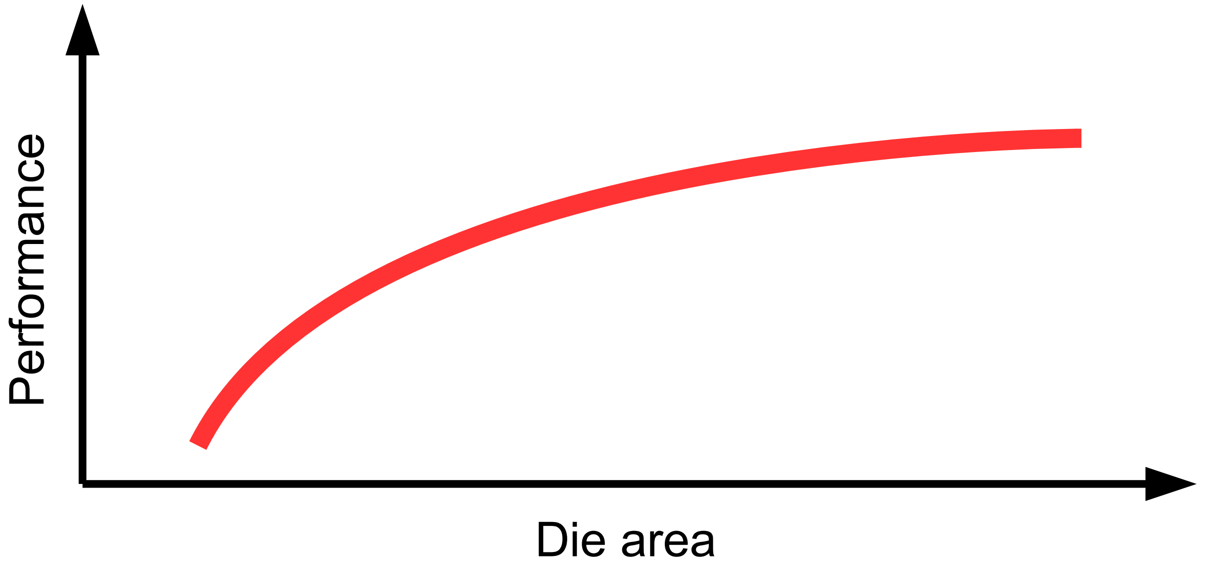
Fraction of chip reachable in one cycle with an 8FO4 clock period



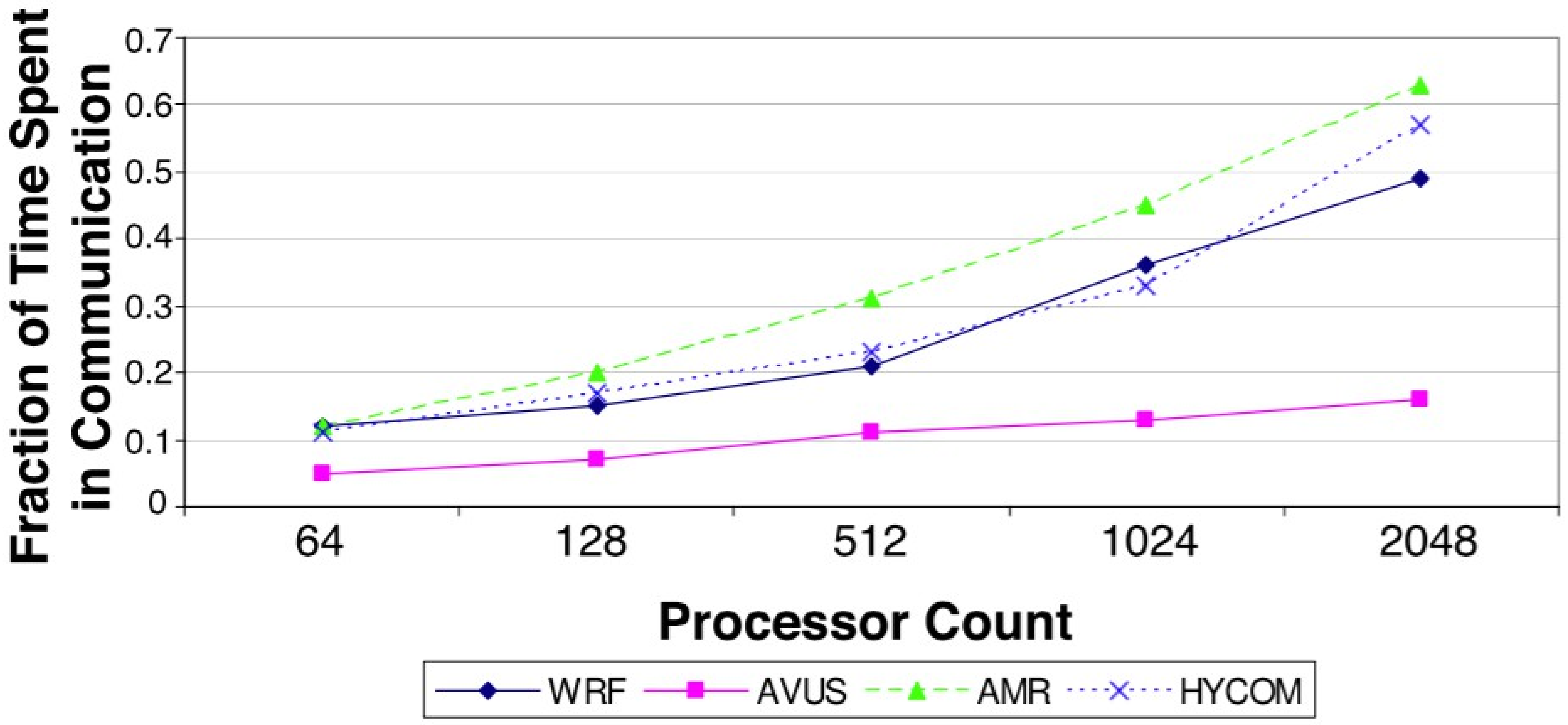
[S. W. Keckler *et al.*, "A wire-delay scalable microprocessor architecture for high performance systems," ISSCC 2003]

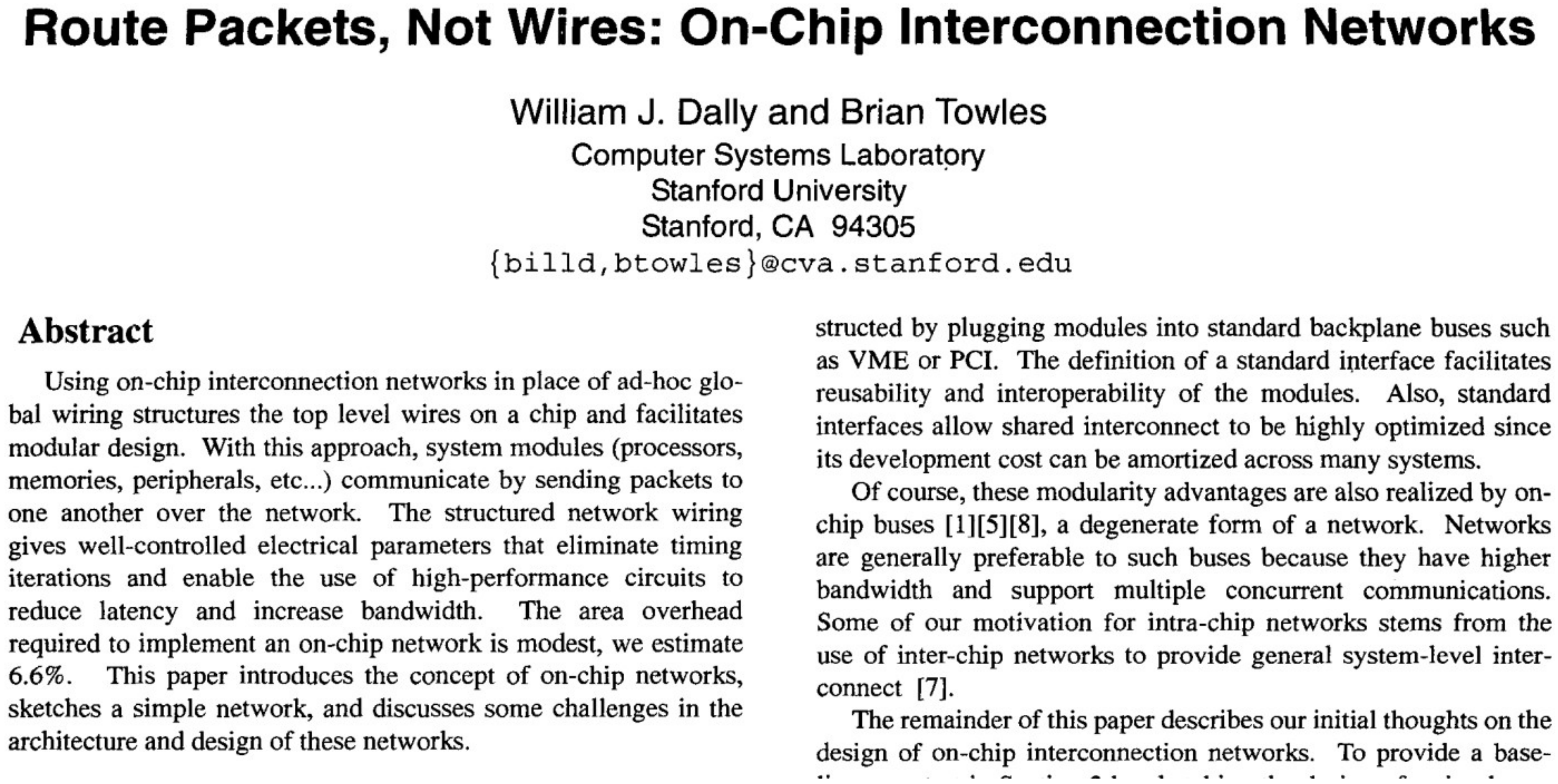
Uniprocessor Architecture Inefficiency

- Pollack's rule
  - New architectures take a lot more area for just a little more performance
  - **...global interconnect is part of this problem!**



Communication Impact





Route Packets, Not Wires: On-Chip Interconnection Networks

William J. Dally and Brian Towles
Computer Systems Laboratory
Stanford University
Stanford, CA 94305
{billd,btowles}@cva.stanford.edu

Abstract

Using on-chip interconnection networks in place of ad-hoc global wiring structures the top level wires on a chip and facilitates modular design. With this approach, system modules (processors, memories, peripherals, etc...) communicate by sending packets to one another over the network. The structured network wiring gives well-controlled electrical parameters that eliminate timing iterations and enable the use of high-performance circuits to reduce latency and increase bandwidth. The area overhead required to implement an on-chip network is modest, we estimate 6.6%. This paper introduces the concept of on-chip networks, sketches a simple network, and discusses some challenges in the architecture and design of these networks.

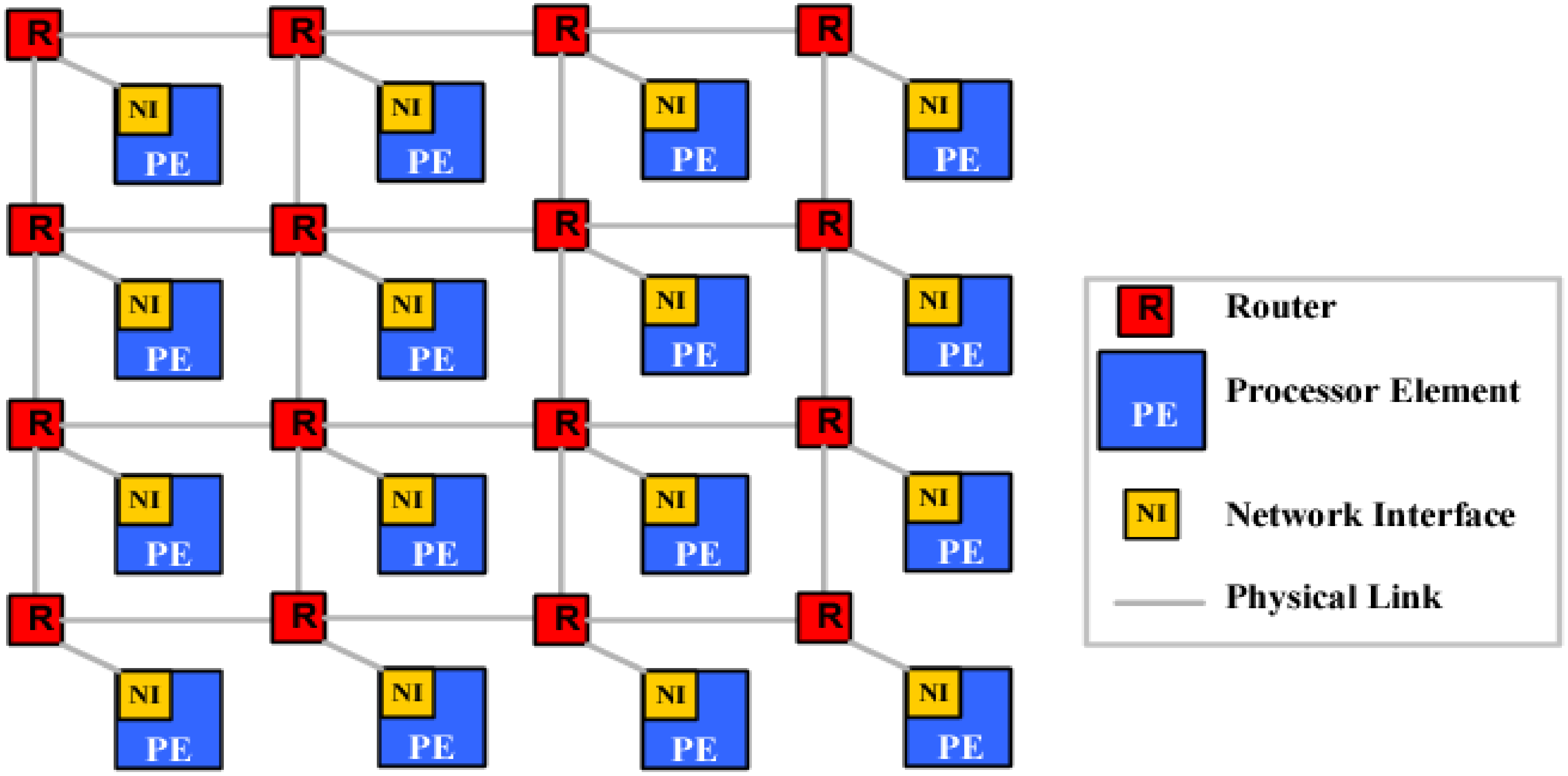
structed by plugging modules into standard backplane buses such as VME or PCI. The definition of a standard interface facilitates reusability and interoperability of the modules. Also, standard interfaces allow shared interconnect to be highly optimized since its development cost can be amortized across many systems.

Of course, these modularity advantages are also realized by on-chip buses [1][5][8], a degenerate form of a network. Networks are generally preferable to such buses because they have higher bandwidth and support multiple concurrent communications. Some of our motivation for intra-chip networks stems from the use of inter-chip networks to provide general system-level interconnect [7].

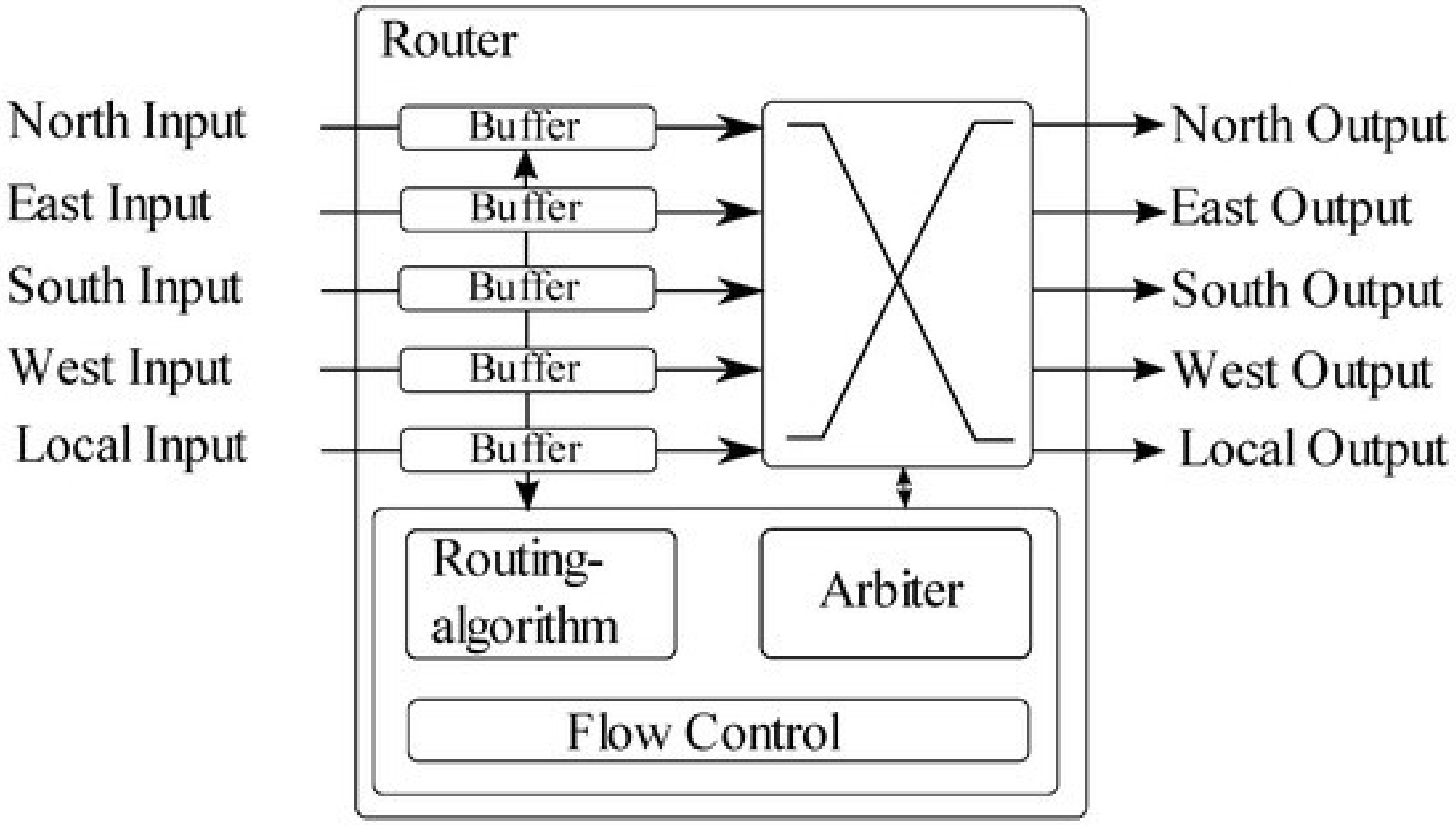
The remainder of this paper describes our initial thoughts on the design of on-chip interconnection networks. To provide a base-

[W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," DAC 2001]

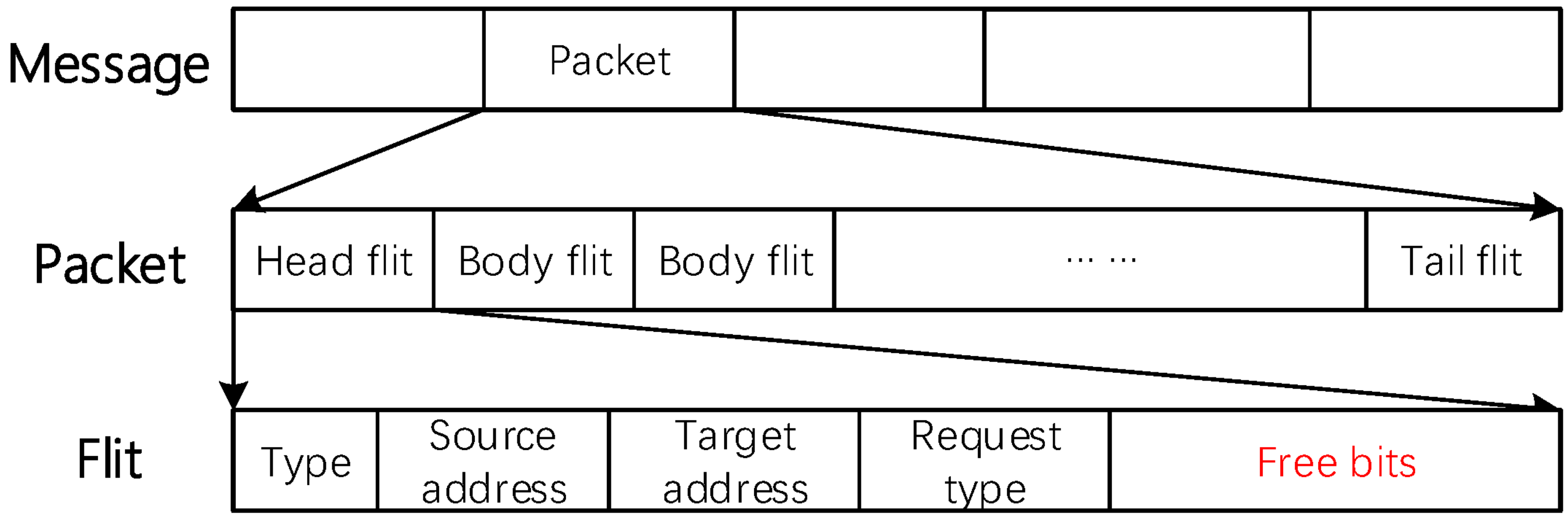
Network-on-Chip Paradigm



NoC Router Architecture



NoC Packet Structure



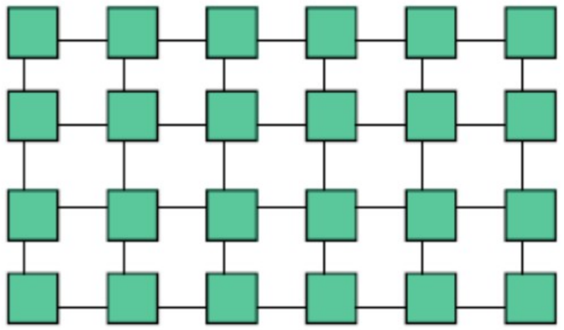
Factors Affecting Perfomance

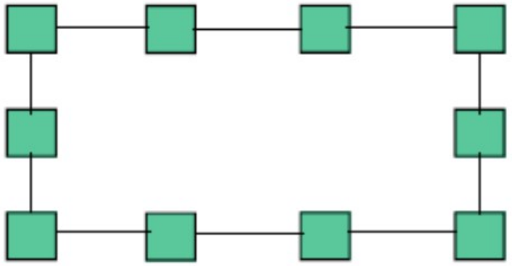
- Topology
- Routing Technique
- Flow Control
- Router Architecture
- Traffic Pattern

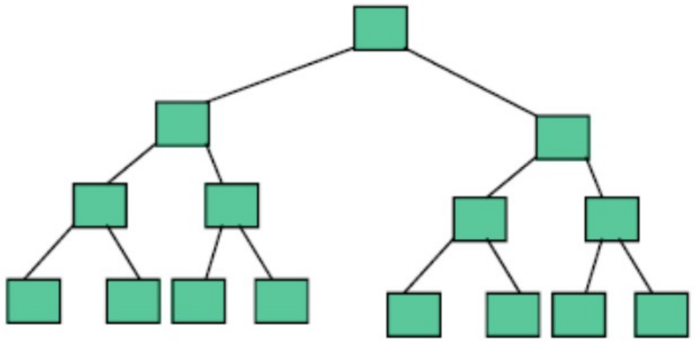
Factors Affecting Perfomance

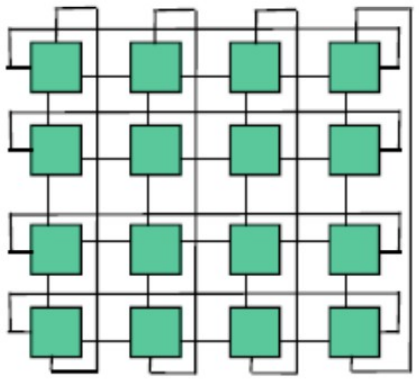
- **Topology**
- Routing Technique
- Flow Control
- Router Architecture
- Traffic Pattern

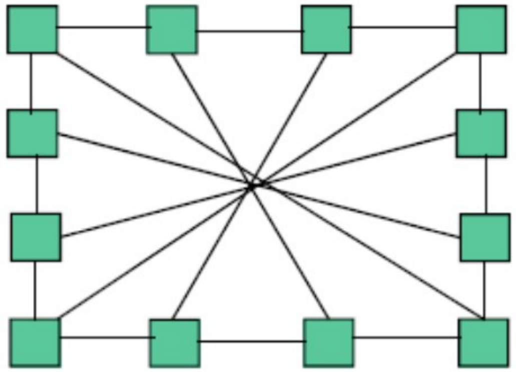
Network Topologies

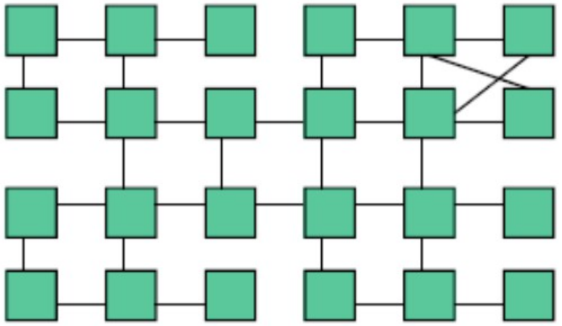

MXN Mesh Network


Ring Network


Binary Tree Network

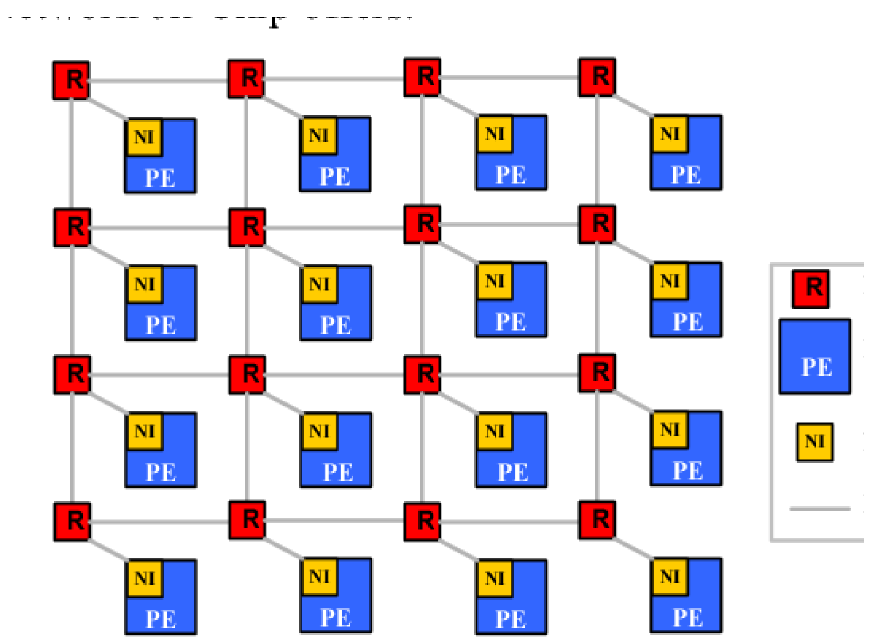

Torus Network


Spidergon network


Application specific topology

Latency

- The ***latency*** of a network is the time required for a packet to traverse the network
  - ➔From the time the head of the packet arrives at the input port to the time the tail of the packet departs the output port

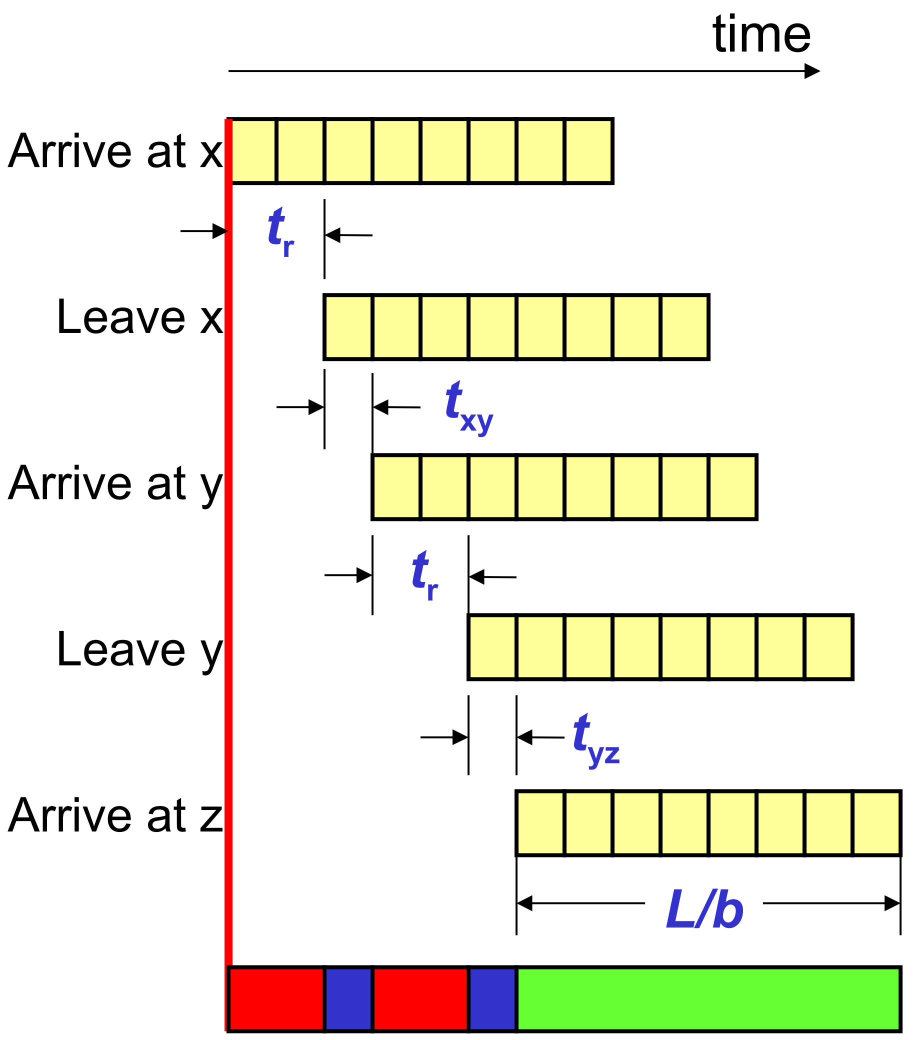


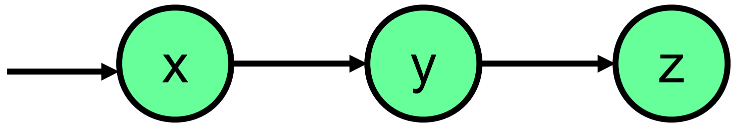
Components of the Latency

- We separate latency, ***T***, into two components
  - ➔ ***Head latency*** (T_h): time required for the head to traverse the network
  - ➔ ***Serialization latency*** (T_s): time for a packet of length ***L*** to cross a channel with bandwidth ***b***

$$T = T_h + T_s = T_h + \frac{L}{b}$$

Packet Propagation





🟥 Routing delay

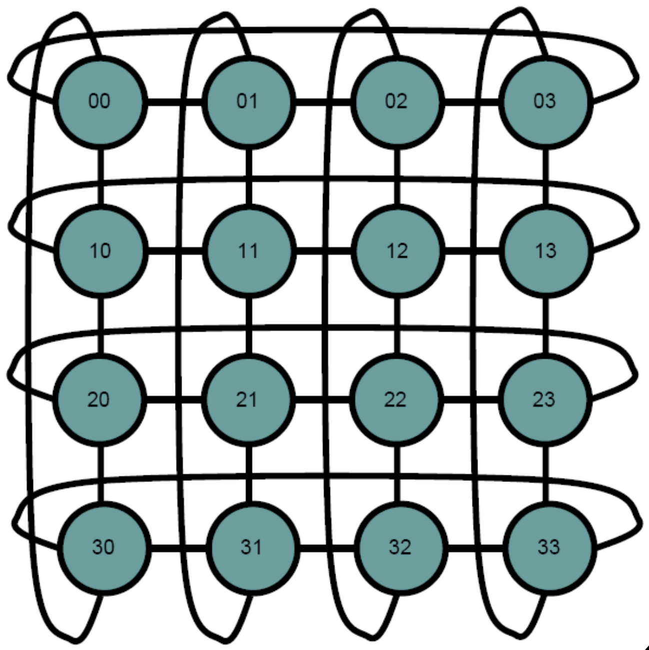
🟦 Link latency

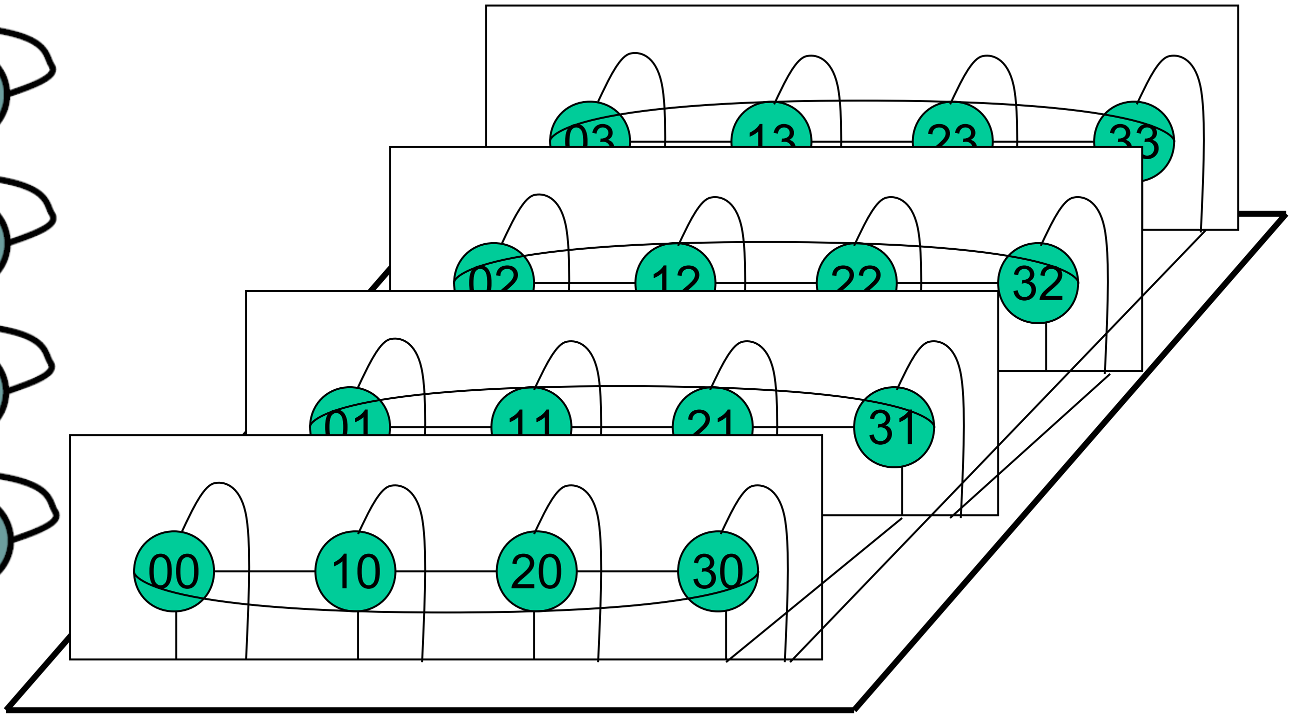
🟩 Serialization latency

Case Study

- A good topology exploits characteristics of the available packaging technology to meet *bandwidth* and *latency* requirements of the application
- To maximize bandwidth a topology should saturate the *bisection bandwidth*

Bandwidth Analysis (Torus)





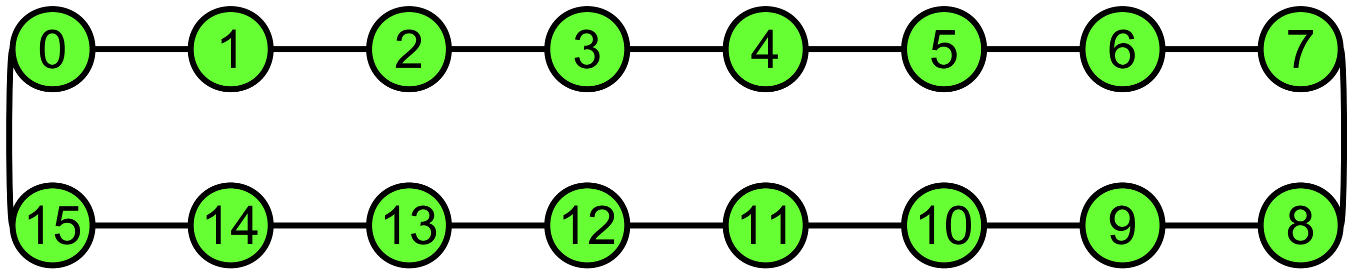
Assume: 256 signals @ 1Gbits/s

Bisection bandwidth 256 Gbits/s

Bandwidth Analysis (Torus)

- 16 unidirectional channels cross the mid-point of the topology
- To saturate the bisection of 256 signals
  - ➔ Each channel crossing the bisection should be 256/16 = 16 signals wide
- Constraints
  - ➔ Each node packaged on a IC
    - ✓ Limited number of I/O pins (*e.g.*, 128)
    - ✓ 8 channels per node → 8x16=128 pins → OK

Bandwidth Analysis (Ring)



- 4 unidirectional channels cross the mid-point of the topology
- To saturate the bisection of 256 signals
  - ➔ Each channel crossing the bisection should be 256/4 = 64 signals wide
- Constraints
  - ➔ Each node packaged on a IC
    - ✓ Limited number of I/O pins (*e.g.*, 128)
    - ✓ 4 channels per node → 4x64=256 pins → INVALID
  - ➔ With identical technology constraints, the ring provides only half the bandwidth of the torus

Delay Analysis

- The application requires only 16Gbits/s
  - …but also minimum latency
- The application uses long 4,096-bit packets
- Suppose *random* traffic
  - Average hop count
    - ✓ Torus = 2
    - ✓ Ring = 4
- Channel size
  - Torus = 16 bits
  - Ring = 32 bits

Delay Analysis

- Serialization latency (channel speed 1GHz)
  - ➔Torus = 4,096/16 * 1ns = 256 ns
  - ➔Ring = 4,096/32 * 1ns = 128 ns
- Latency assuming 20ns hop delay
  - ➔Torus = 256 + 20*2 = 296 ns
  - ➔Ring = 128 + 20*4 = 208 ns
- No one topology is optimal for all applications
  - ➔Different topologies are appropriate for different constraints and requirements

Factors Affecting Perfomance

- Topology
- Routing Technique
- **Flow Control**
- Router Architecture
- Traffic Pattern

Flow Control

- *Flow Control* determines how the resources of a network, such as channel bandwidth and buffer capacity are allocated to packets traversing a network
- Goal is to use resources as efficient as possible to allow a high throughput
- An efficient flow control is a prerequisite to achieve a good network performance

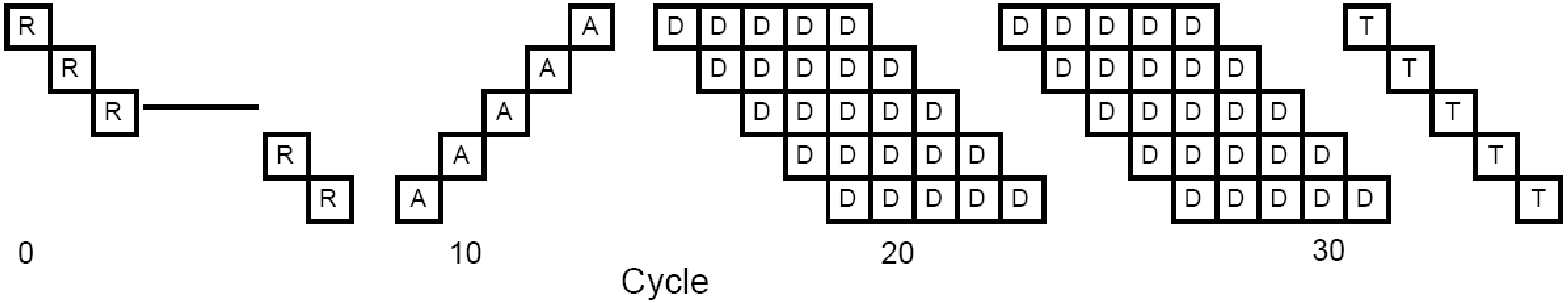
Flow Control

- Flow Control can be viewed as a problem of
  - ➔ Resource allocation
  - ➔ Contention resolution
- Resources in form of channels, buffers and state must be allocated to each packet
- If two packets compete for the same channel flow control can only assign the channel to one packet, but must also deal with the other packet

Flow Control

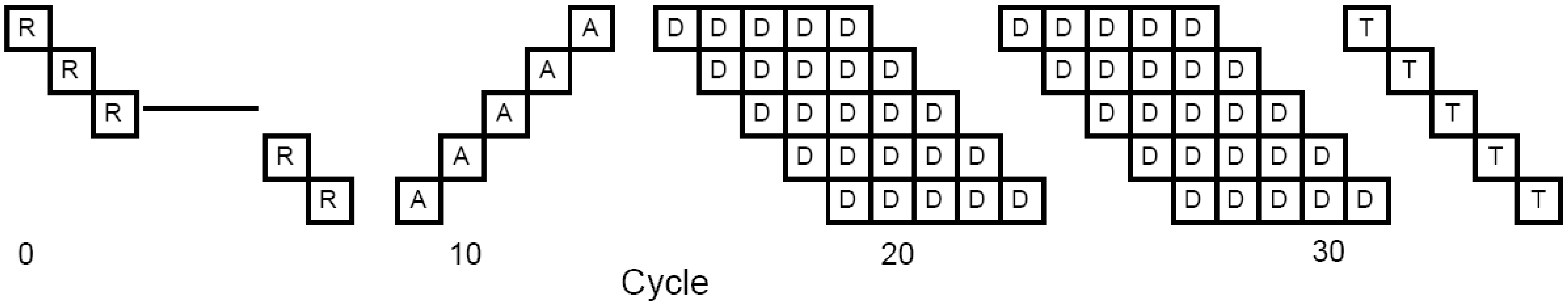
- Flow Control can be divided into
  - ➔ Bufferless flow control
    - ✓ Packets are either dropped or misrouted
  - ➔ Buffered flow control
    - ✓ Packets that cannot be routed via the desired channel are stored in buffers

Circuit Switching



- Circuit-Switching is a bufferless flow control, where several channels are reserved to form a circuit
- A request (*R*) propagates from source to destination, which is answered by an acknowledgement (*A*)
- Then data is sent (here two five flit packets (*D*)) and a tail flit (*T*) is sent to deallocate the channels

Circuit Switching



- Circuit-switching does not suffer from dropping or misrouting packets
- However there are two weaknesses
  - ➔ High latency: $T_0 = 3\,H\,t_r + L/b$ (ignoring wire latency)
  - ➔ Low throughput, since channel is used to a large fraction of time for signaling and not for delivery of the payload

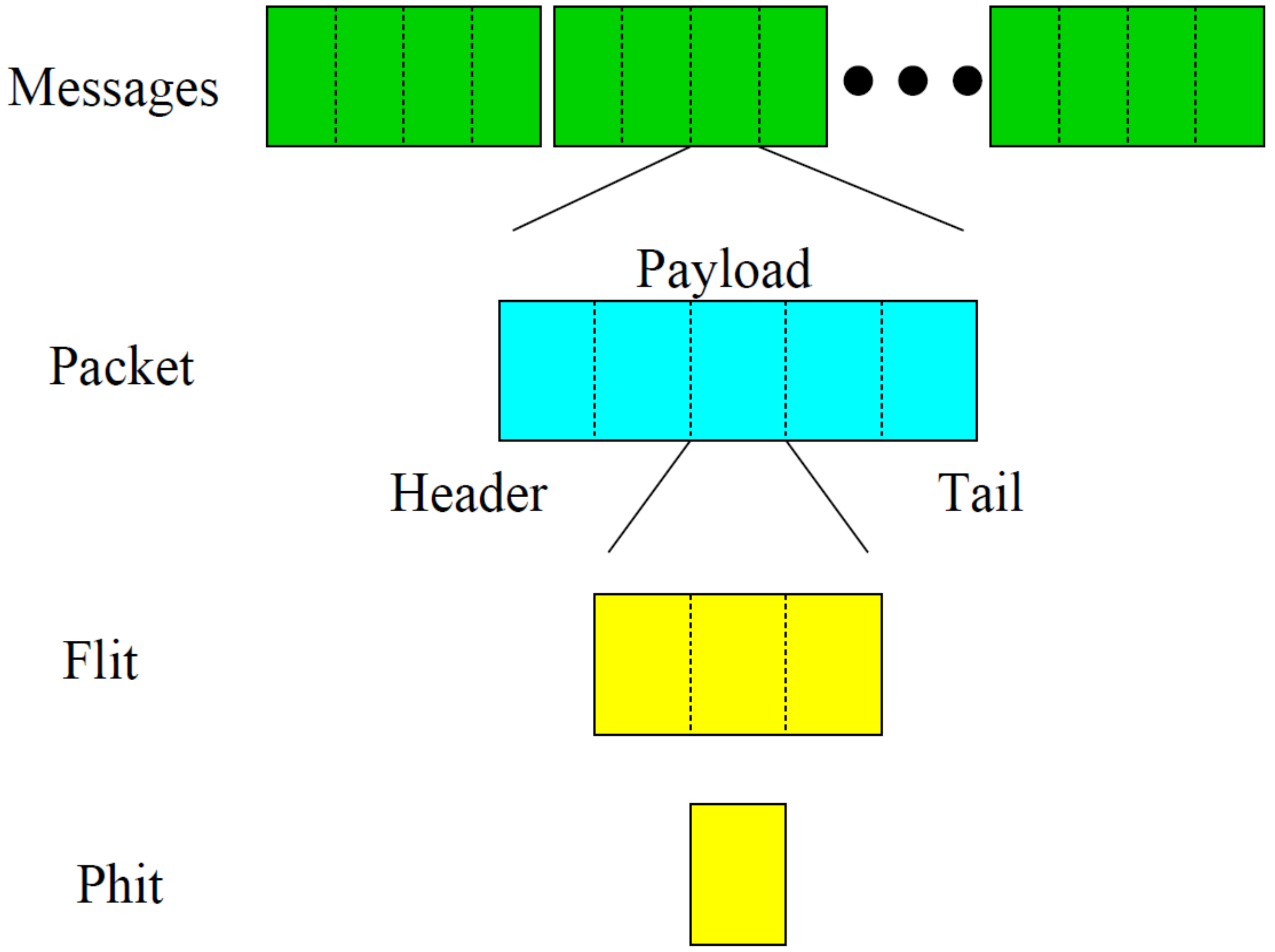
Buffered Flow Control

- More efficient flow control can be achieved by adding buffers
  - ➔With sufficient buffers packets do not need to be misrouted or dropped, since packets can wait for the outgoing channel to be ready

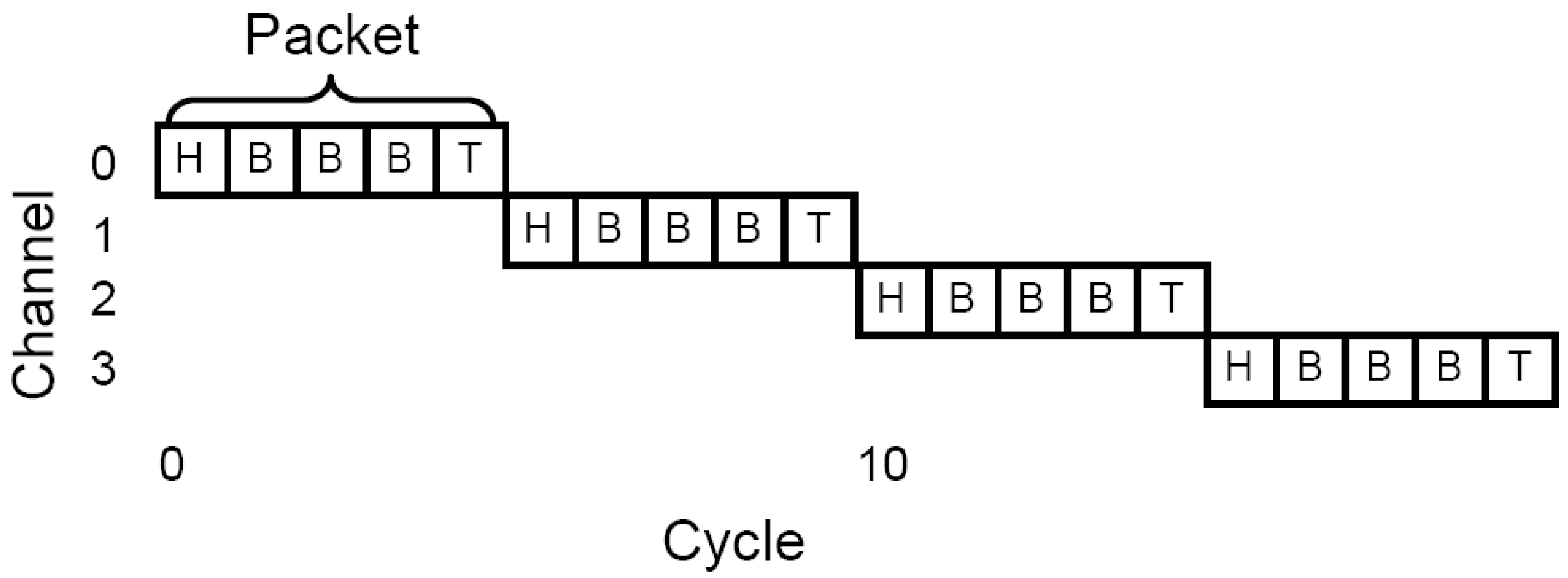
Buffered Flow Control

- Two main approaches
  - ➔ Packet-Buffer Flow Control
    - ✓ Store-And-Forward
    - ✓ Cut-Through
  - ➔ Flit-Buffer Flow Control
    - ✓ Wormhole Flow Control
    - ✓ Virtual Channel Flow Control

Data Units



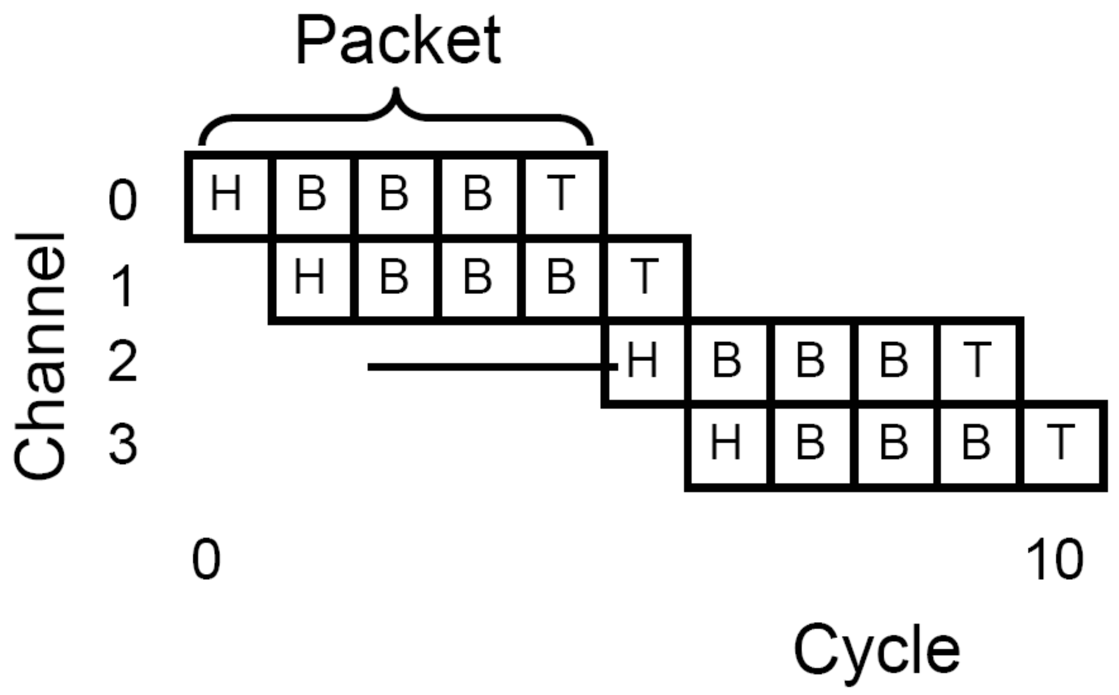
Store and Forward Flow Control



- Each node along a route waits until a packet is completely received (stored) and then the packet is forwarded to the next node
- Two resources are needed
  - ➔Packet-sized buffer in the switch
  - ➔Exclusive use of the outgoing channel

$$T_0 = H\,(t_r + L/b)$$

Cut-Through Flow Control



- Transmission on the next channel starts directly when the new header flit is received (otherwise it behaves like Store-Forward)
  - ➔ Channel is released after tail flit

$$T_0 = H\, t_r + L/b$$

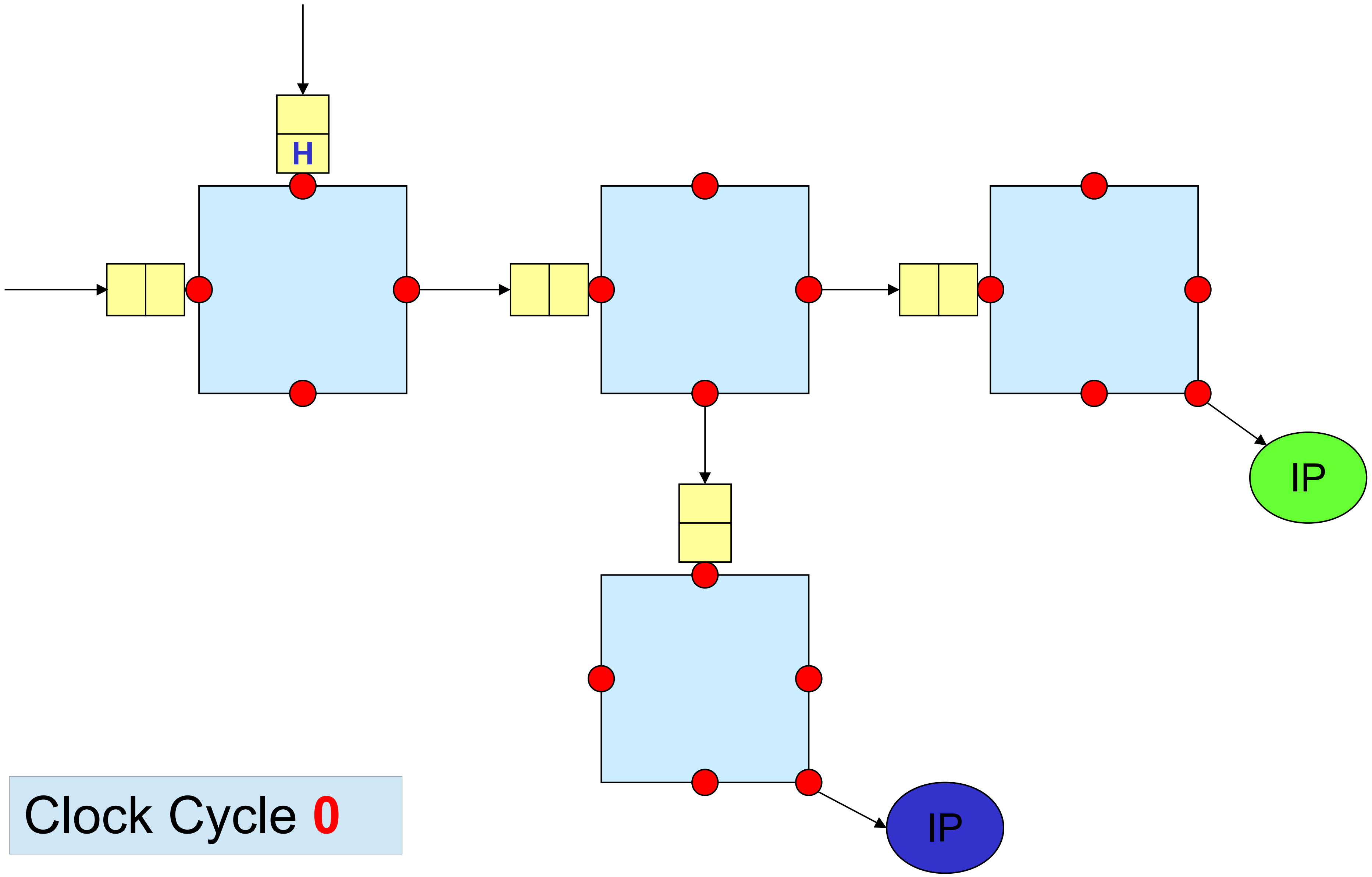
Cut-Through Flow Control

- Shortcomings
  - ➔ Very inefficient use of buffer space
    - ✓ As buffers are allocated in units of packets
    - ✓ Often we need multiple indipendent buffer sets to reduce blocking or provide deadlock avoidance
  - ➔ By allocating buffers in units of packets → contention latency is increased
    - ✓ E.g., High-priority packet colliding with a low-priority packet
      - – Must wait the entire low-priority packet to be transmitted before it can acquire the channel

Wormhole Flow Control

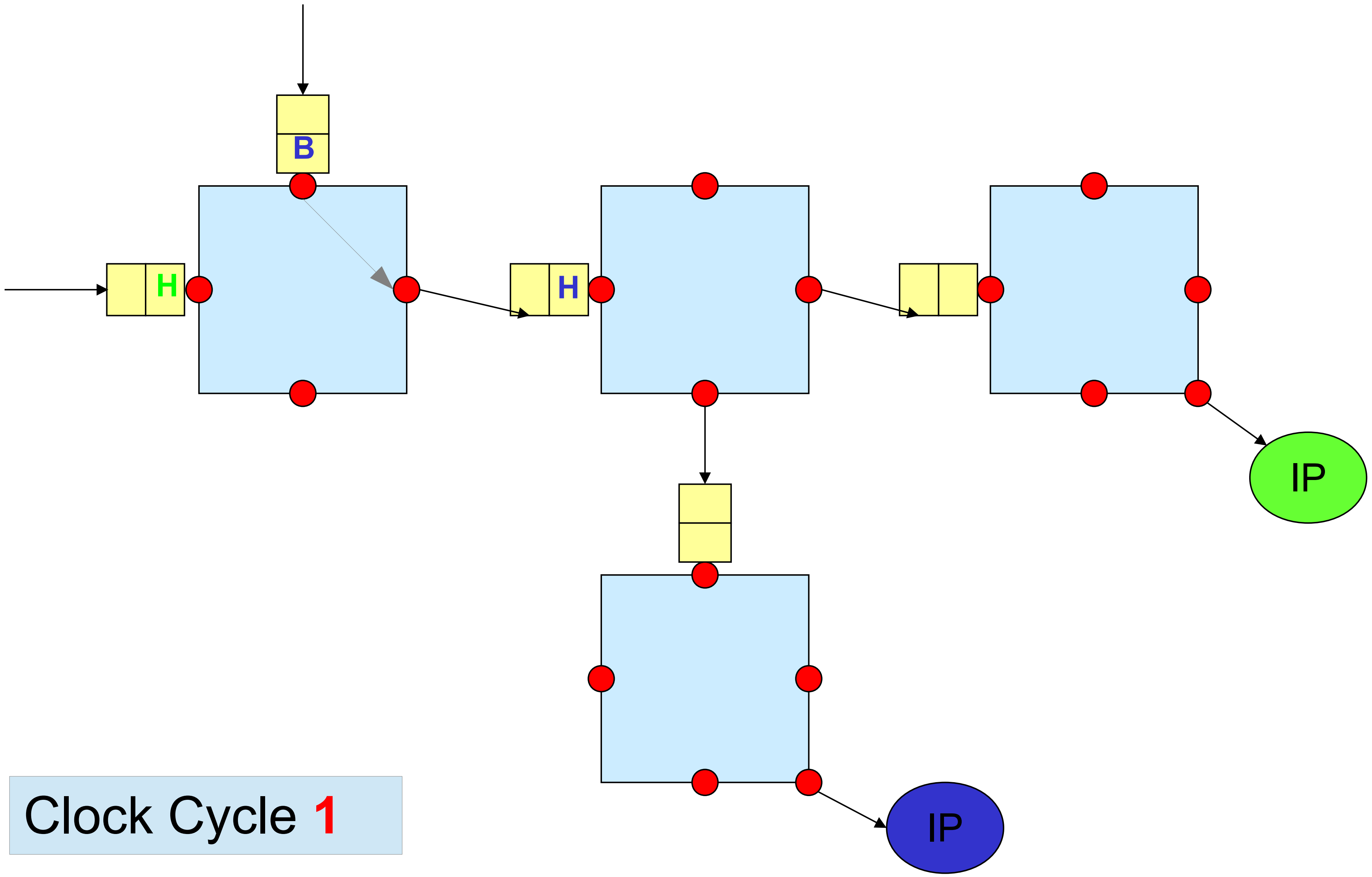
- Wormhole flow control operates like cut-through, but with channel and buffers allocated to flits rather than packets
- Three resources are needed
  - ➔ A virtual channel for the packet
    - ✓ Body flits of a packet use the VC acquired by the head flit
  - ➔ One flit buffer
  - ➔ One flit channel bandwidth

Wormhole - Example



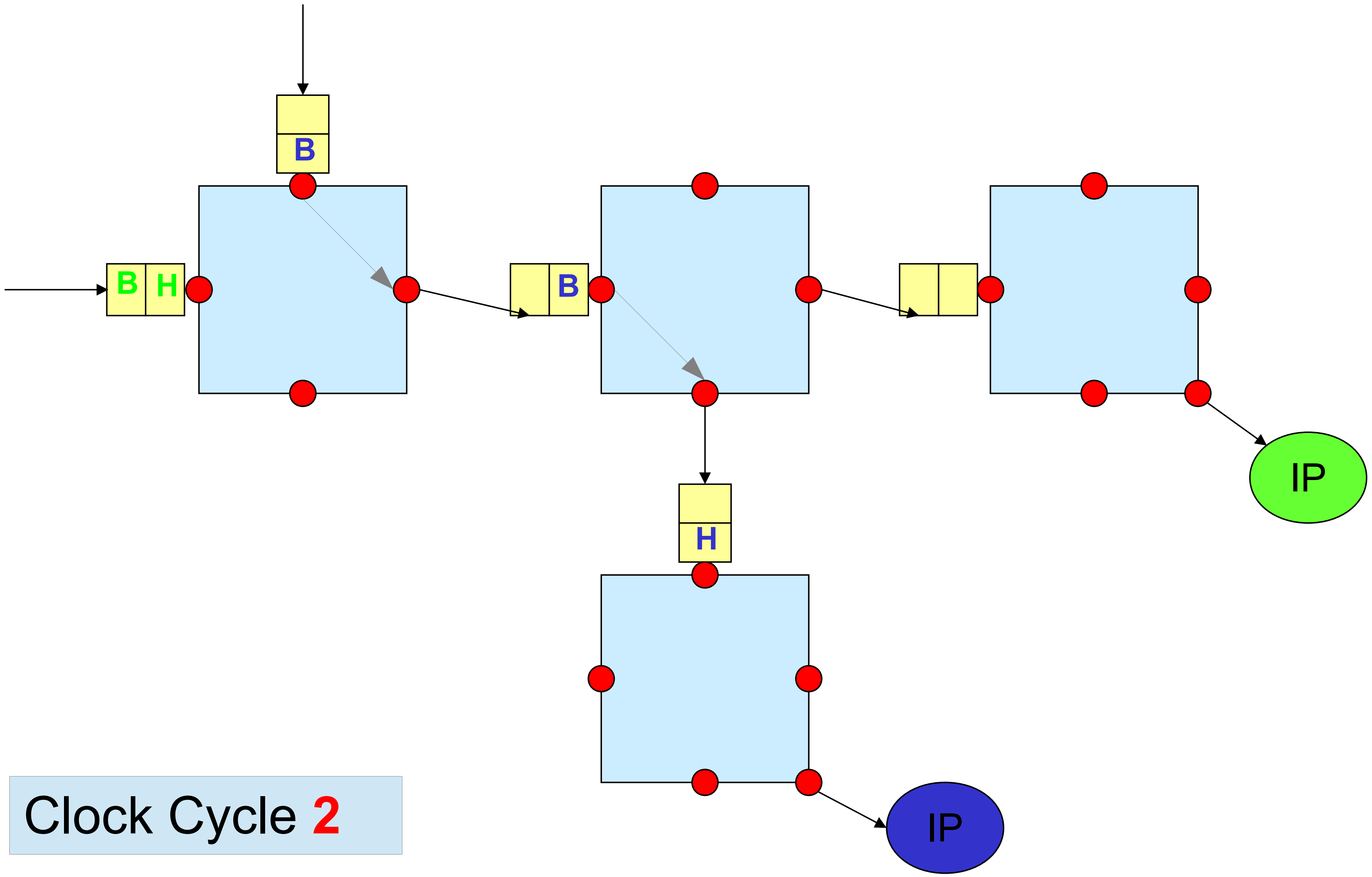
Clock Cycle **0**

Wormhole - Example



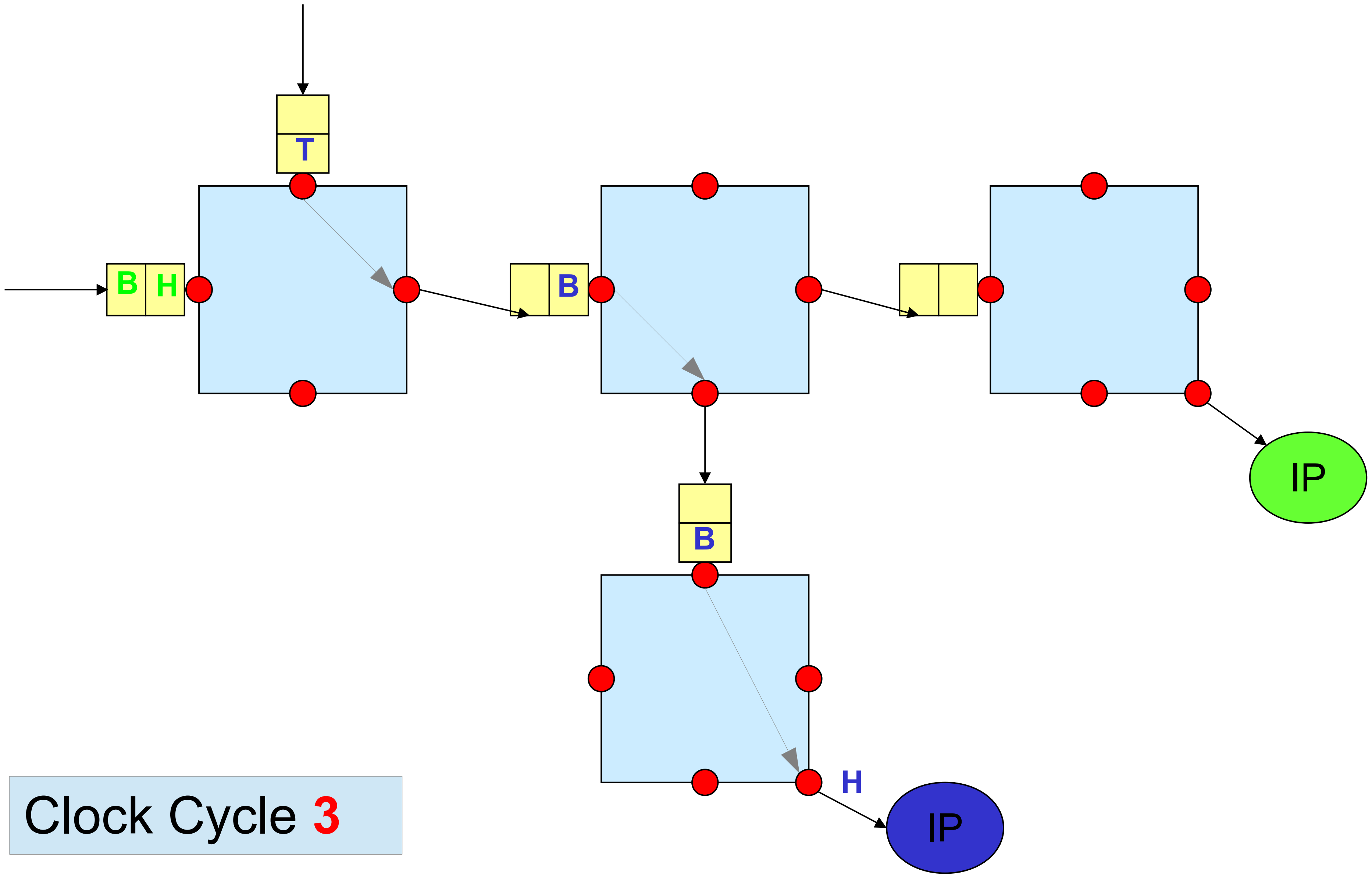
Clock Cycle 1

Wormhole - Example



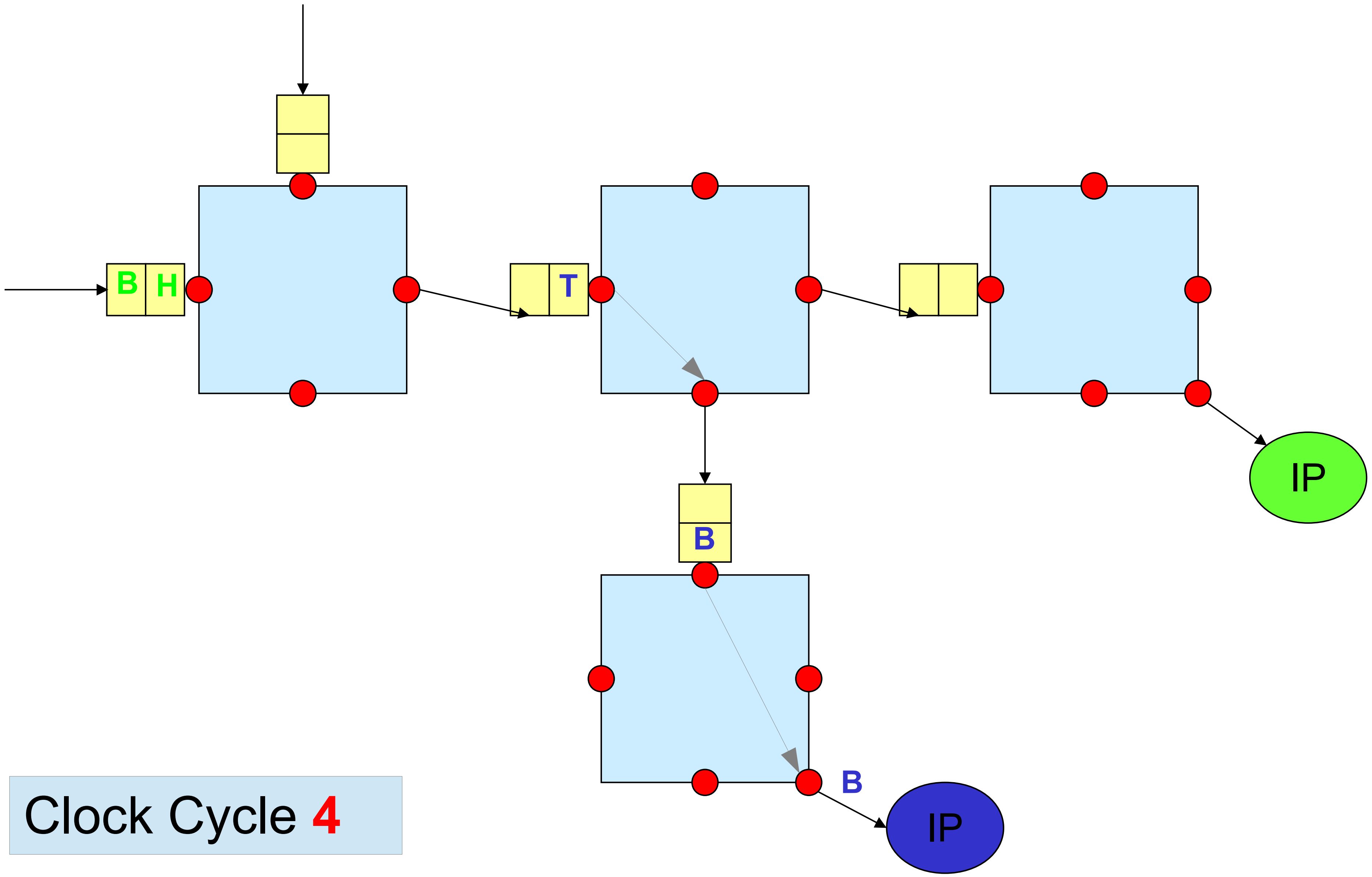
Clock Cycle **2**

Wormhole - Example



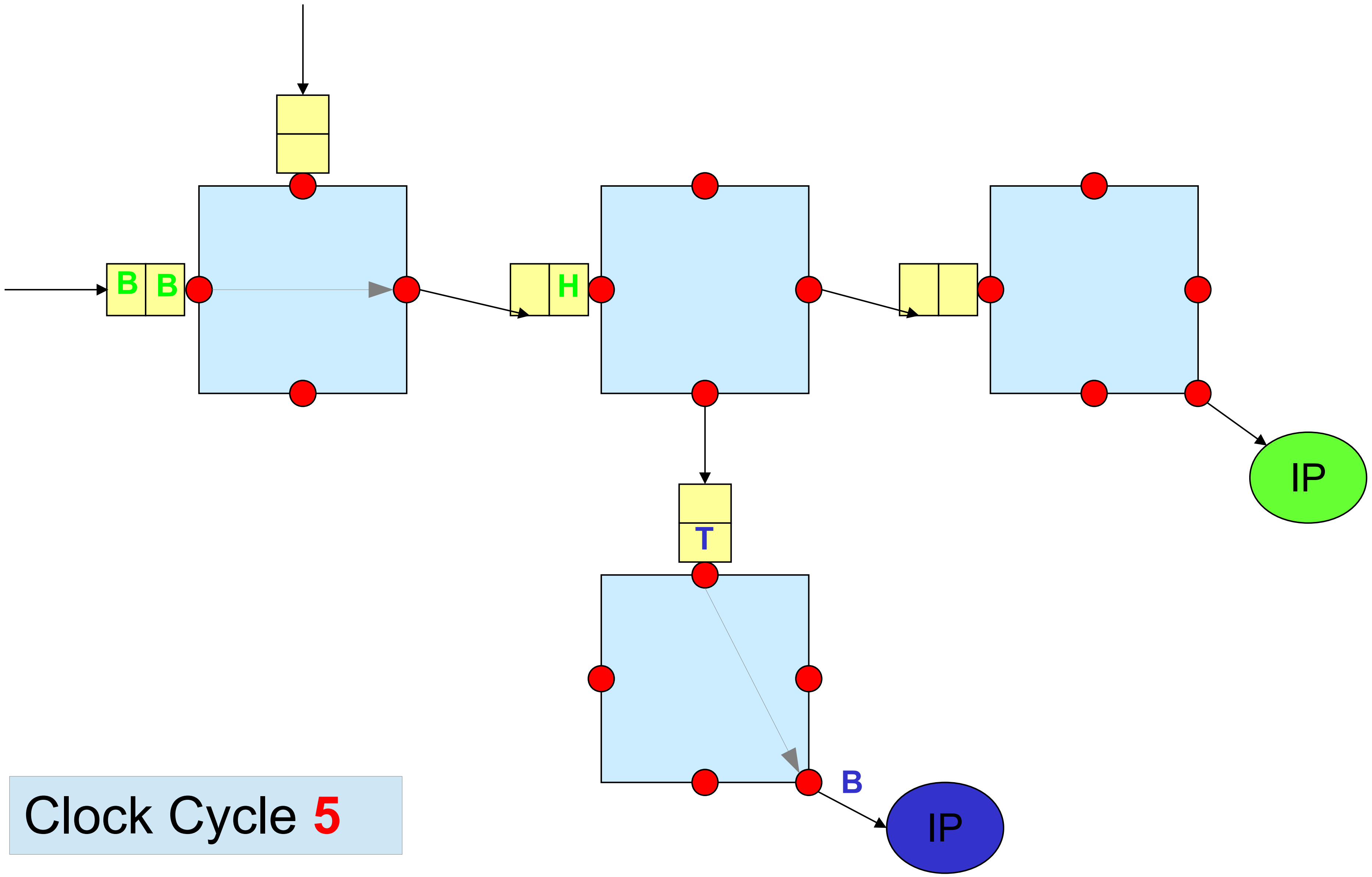
Clock Cycle **3**

Wormhole - Example



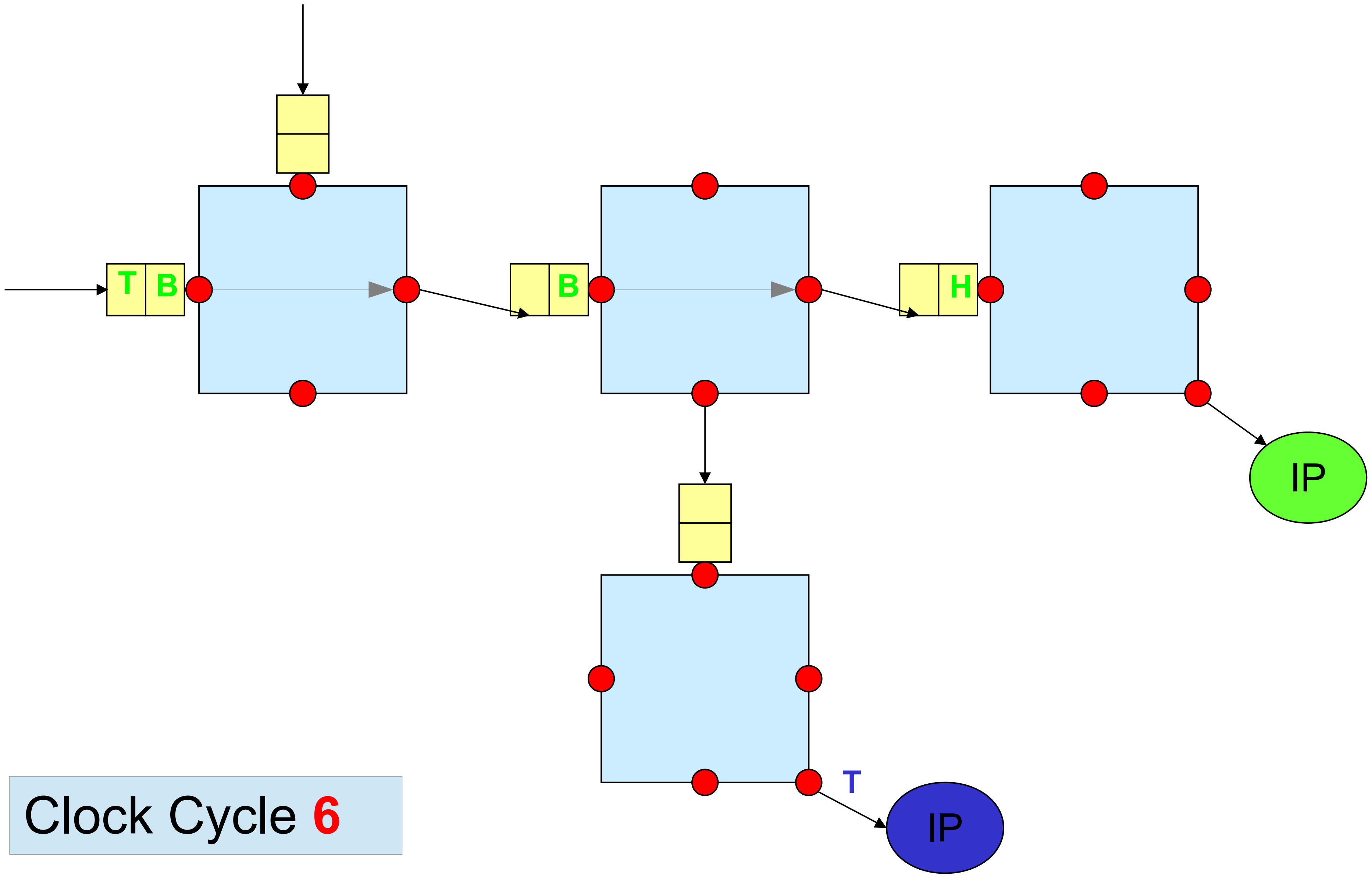
Clock Cycle **4**

Wormhole - Example



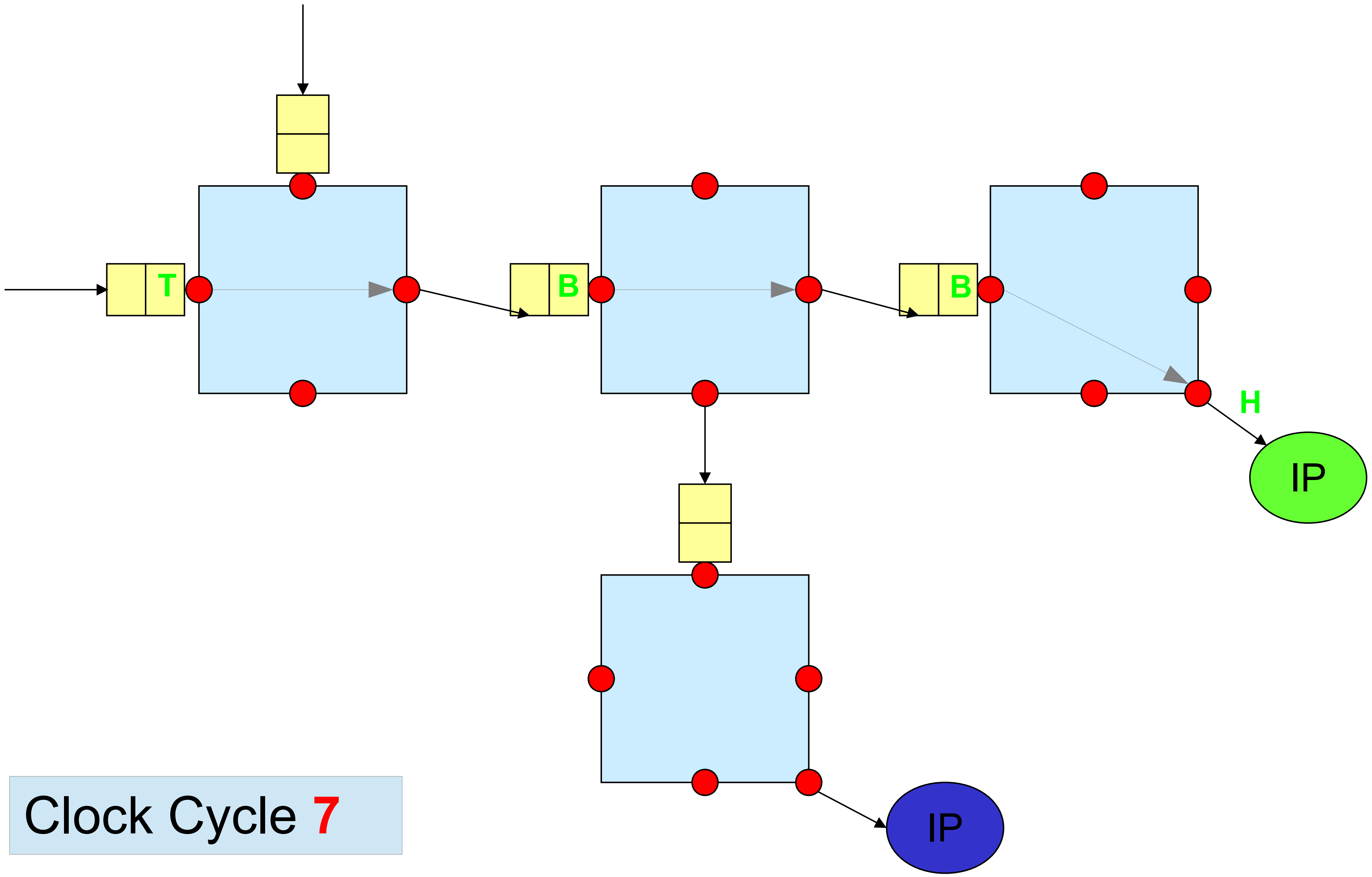
Clock Cycle **5**

Wormhole - Example



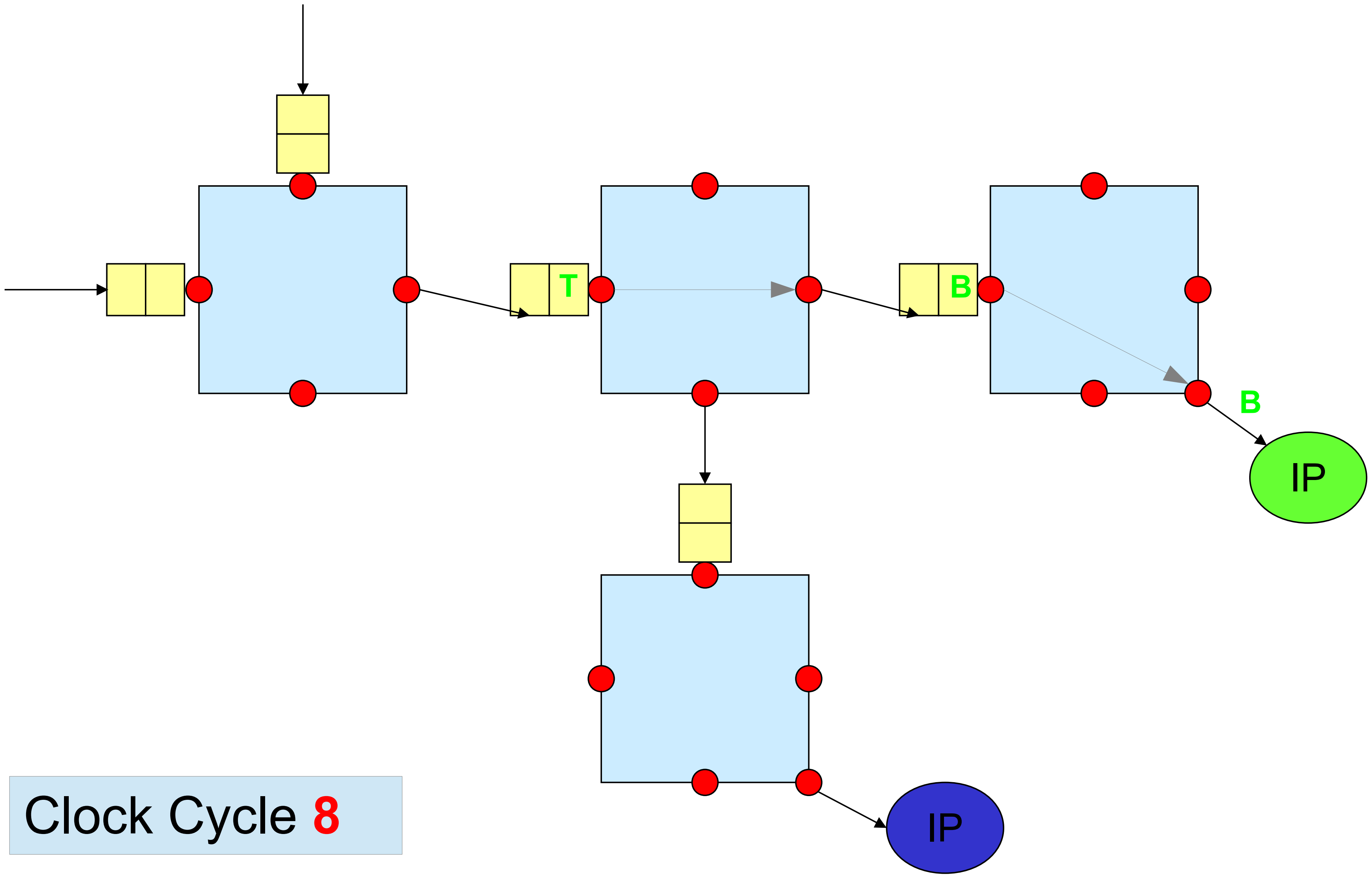
Clock Cycle **6**

Wormhole - Example



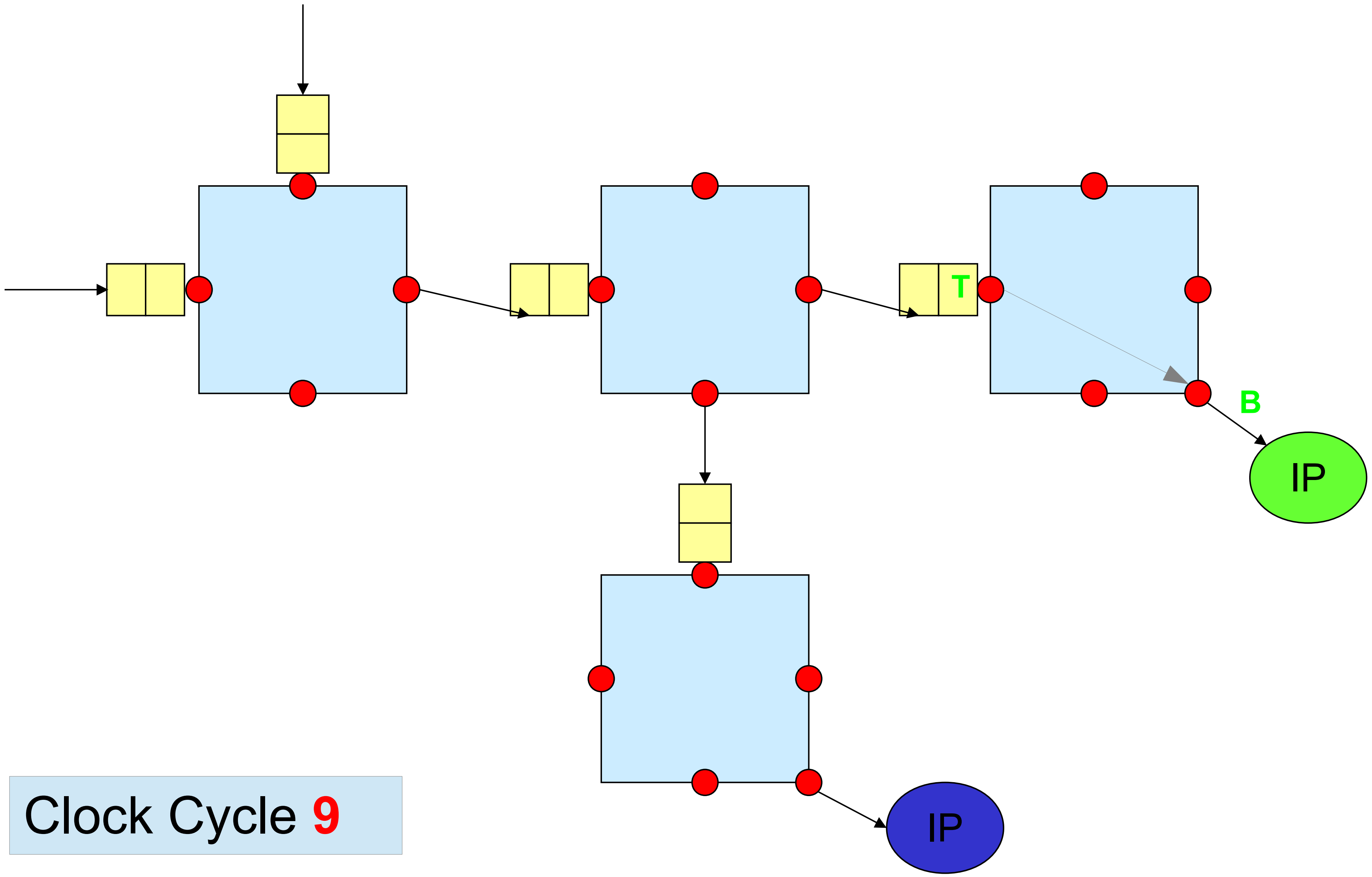
Clock Cycle 7

Wormhole - Example



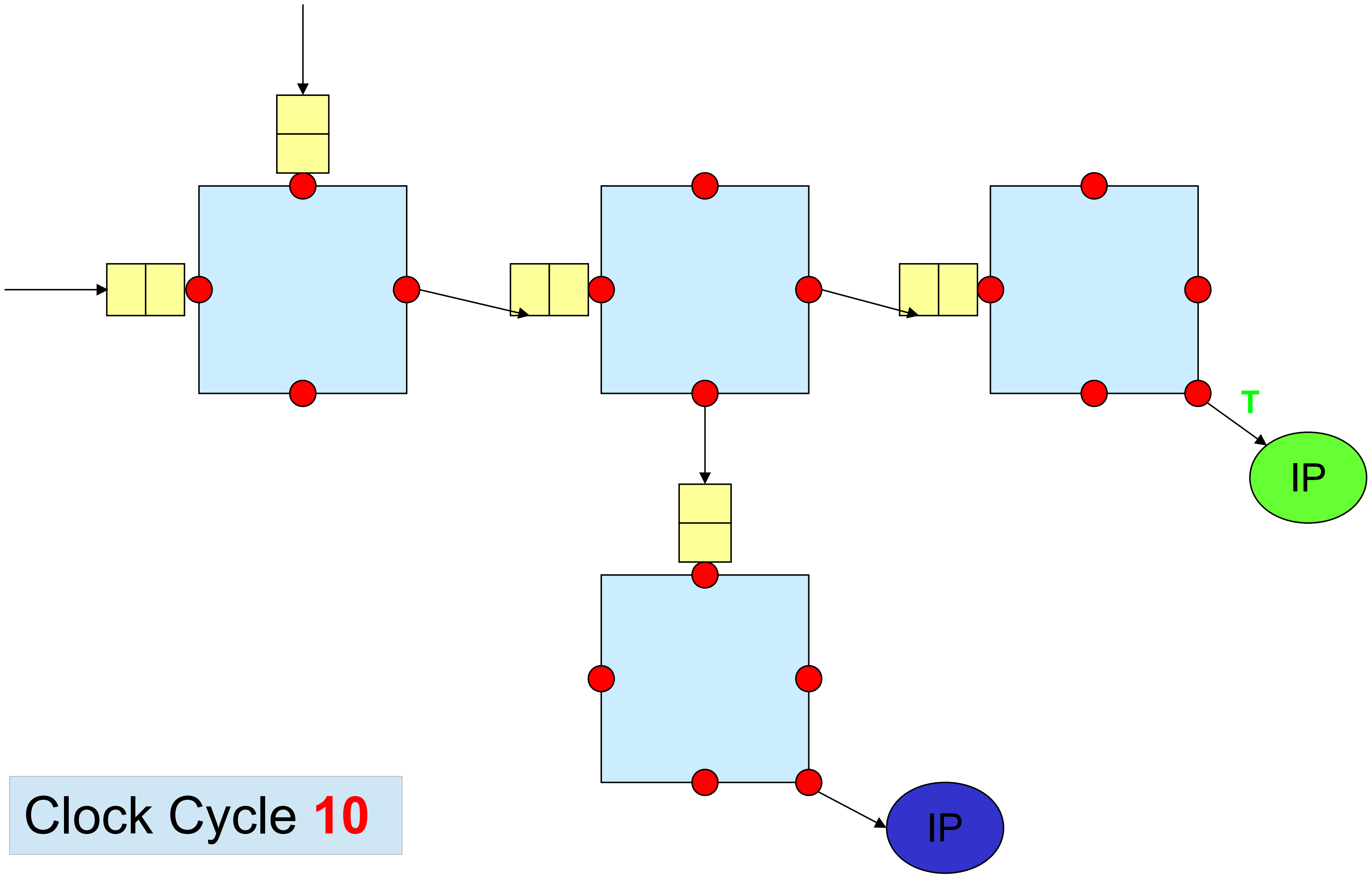
Clock Cycle **8**

Wormhole - Example



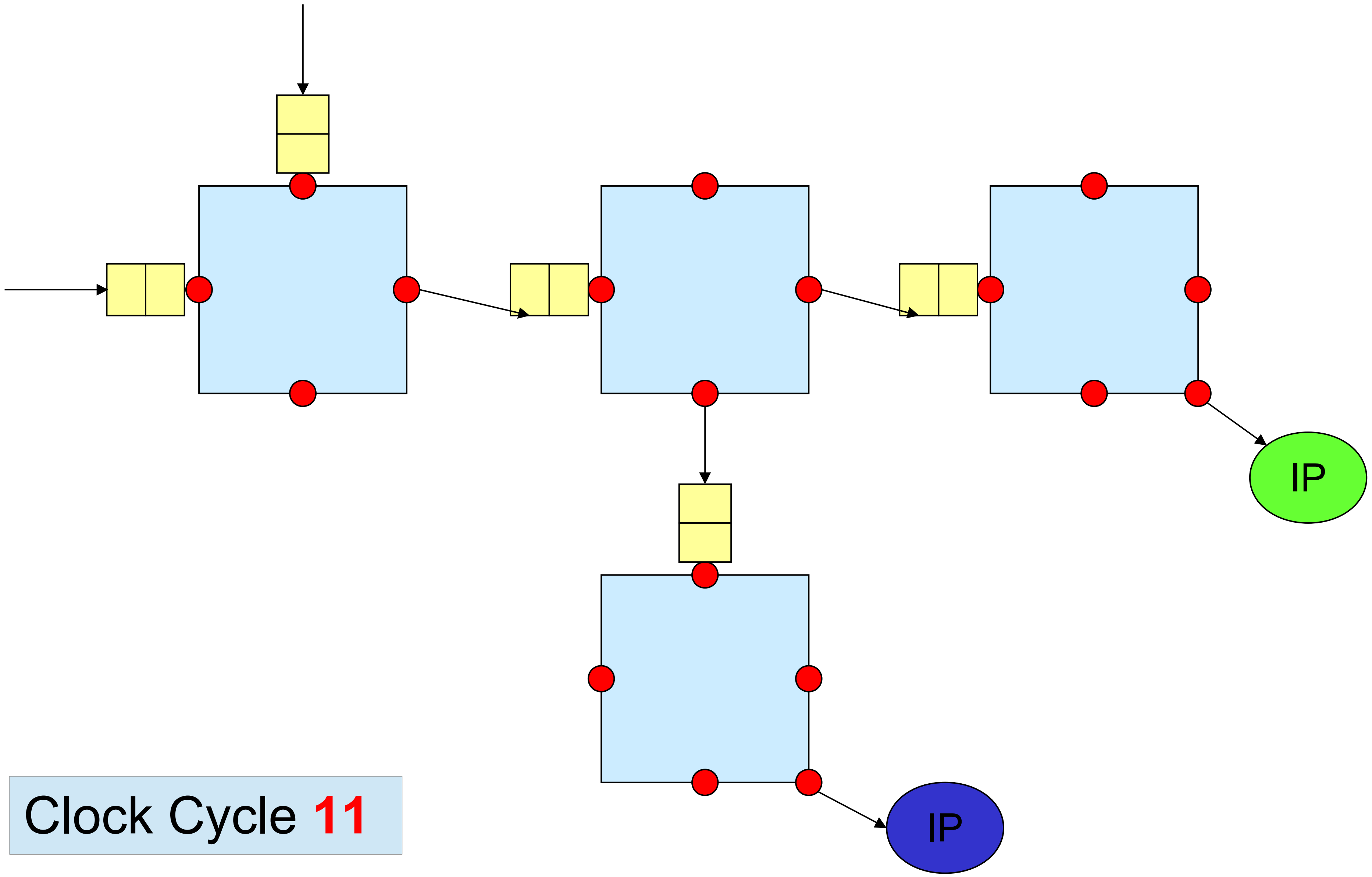
Clock Cycle **9**

Wormhole - Example



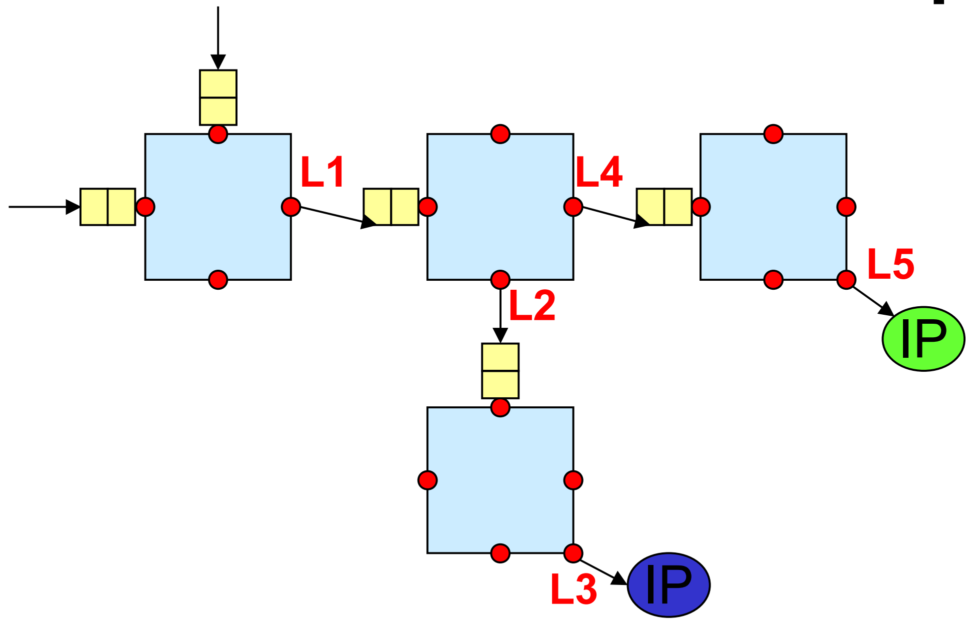
Clock Cycle **10**

Wormhole - Example



Clock Cycle **11**

Wormhole - Example



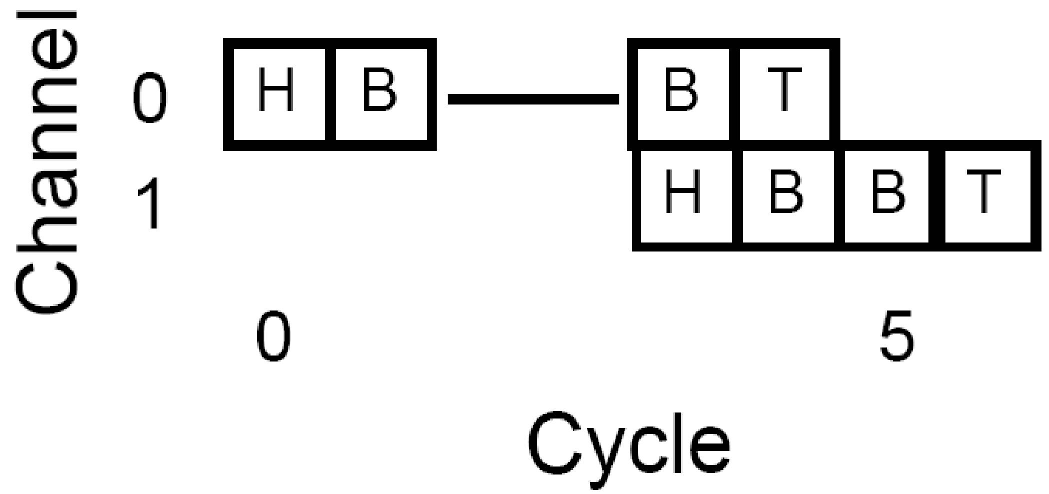
CC	0	1	2	3	4	5	6	7	8	9	10	11
Blue	Injected	L1	L1, L2	L1, L2, L3	L1, L2, L3	L2, L3	L3	Drained				
Green		Injected				L1	L1, L4	L1, L4, L5	L1, L4, L5	L4, L5	L5	Drained

- **Blue packet**
  - Injected at CC 0
  - Delivered at CC 7
  - **Latency 7 clock cycles**
- **Green packet**
  - Injected at CC 1
  - Delivered at CC 11
  - **Latency 10 clock cycles**

Wormhole Flow Control

- Comparison to cut-through
  - ➔Wormhole flow control makes far more efficient use of buffer space
  - ➔Throughput maybe less, since wormhole flow control may block a channels mid-packets

Wormhole Flow Control

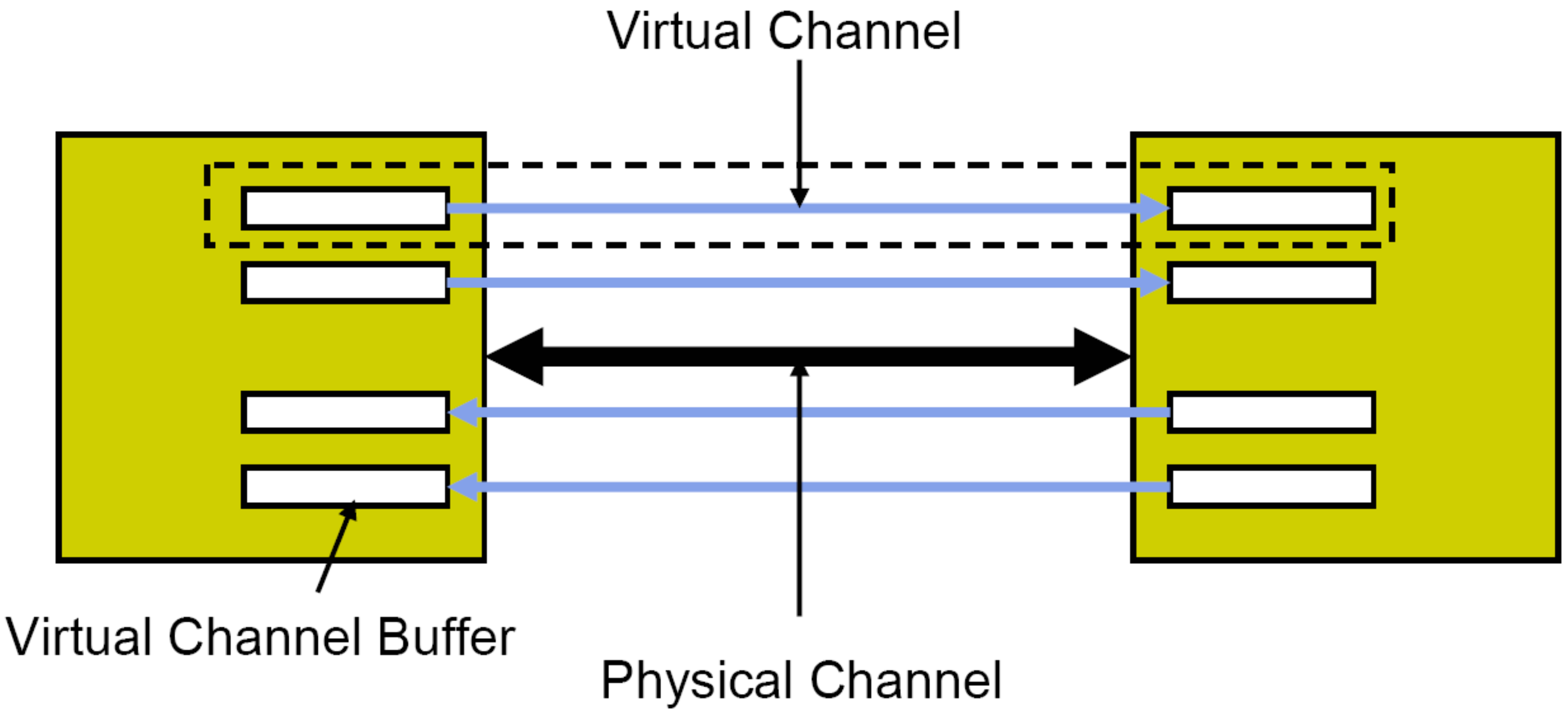


- The main advantage of wormhole to cut-through is that buffers in the routers do not need to be able to hold full packets, but only need to store a number of flits
- This allows to use smaller and faster routers

Virtual Channel Flow Control

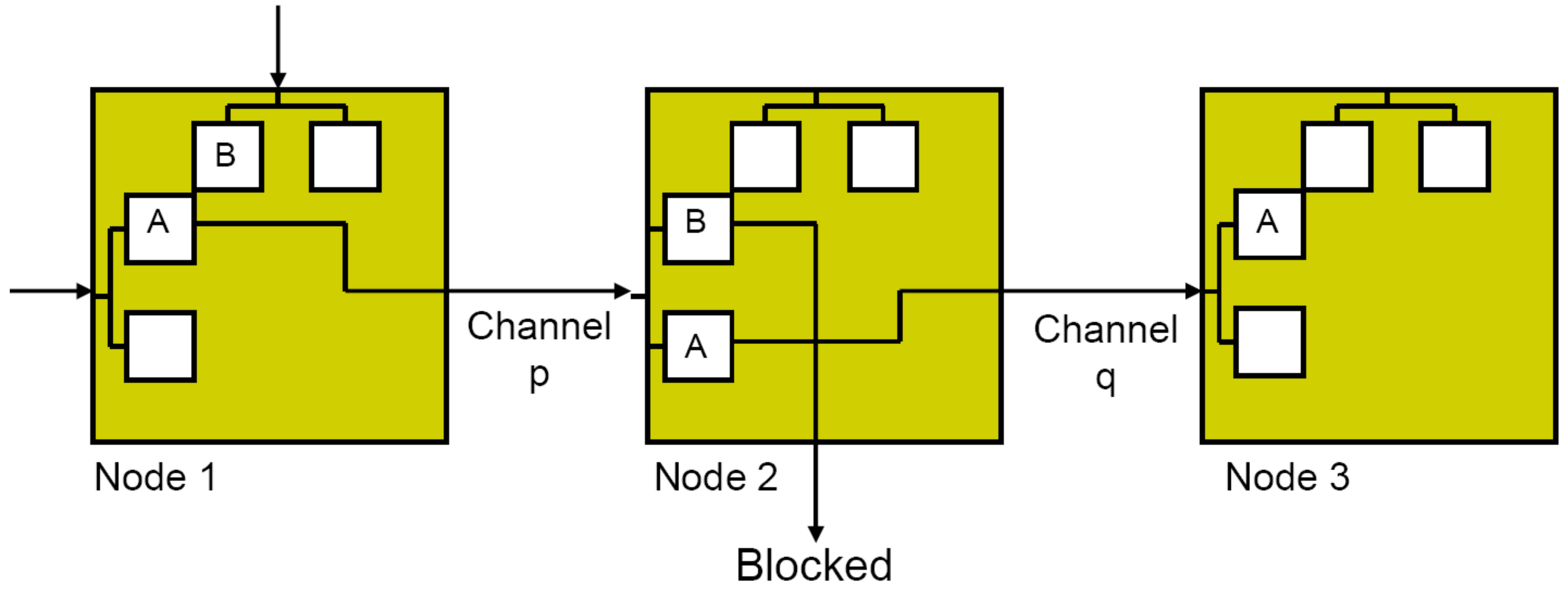
- In virtual channel flow-control several channels are associated with a single physical channel
- This allows to use the bandwidth that otherwise is left idle when a packet blocks the channel
- Unlike wormhole flow control subsequent flits are not guaranteed bandwidth, since they have to compete for bandwidth with other flits

Concept of Virtual Channels



- A physical channel is shared by several virtual channels
- Naturally the speed of each virtual channel connection is reduced

Virtual Channel Flow Control



- There are several virtual channels for each physical channel
- Packet *A* can use a second virtual channel and thus proceed over channel *p* and *q*

Factors Affecting Perfomance

- Topology
- **Routing Technique**
- Flow Control
- Router Architecture
- Traffic Pattern

Generalities

- Routing involves selecting a path from a source node to a destination node in a particular topology

A Good Routing Algorithm...

- Balances load across the network channels
  - Even in the presence of non-uniform traffic patterns
  - The more balanced the channel load, the closer the throughput of the network is to ideal

General Routing Algorithms

- Many routers that have been built and are in use today do a poor job of balancing load
  - The traffic between each pair of nodes follows a single, predetermined path
  - Non-uniform traffic patterns can induce large load misbalances
- Why?

General Routing Algorithms (*cnt'd*)

- Most of these routers have been designed to optimize a second important aspect
  - Short path lengths

Minimal *vs.* Non Minimal Paths

- Keeping path lengths as short as possible
  - Reduces the number of hops
  - ...and then, the overall latency of a message

Minimal *vs.* Non Minimal Paths

- Often, routing minimally does not help in balancing load and maximizing throughput
  - For oblivious routing algorithms, to improve load balance over all traffic patterns, we are forced to increase the average path length of all messages
    - Oblivious routing do not factor the current traffic pattern into the routing algorithm

Oblivious *vs.* Adaptive Routing

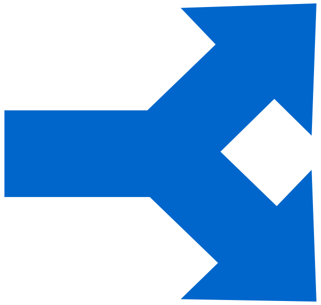
- Best of both worlds
  - Why not adapt to the current traffic conditions?
- Send traffic
  - Minimally for an “easy” traffic pattern such as uniform traffic
  - Non-minimal for “hard” non-uniform traffic patterns

Faults in the Network

- Link or node fails
  - Routing algorithm hardwired into the routers
    - The entire system fails
  - Routing algorithm can be reprogrammed
    - The system can continue to operate

Taxonomy of Routing Algorithms

- Deterministic
- Oblivious
- Adaptive



- Minimal
- Non-minimal

Deterministic Routing Algorithms

- Always choose the same path between two nodes
  - Easy to implement and to make deadlock free
  - Do not use path diversity and thus bad on load balancing

Oblivious Routing Algorithms

- Always choose a route without knowing about the state of the networks state
  - All random algorithms are oblivious algorithms
  - All deterministic algorithms are oblivious algorithms

Adaptive Routing Algorithms

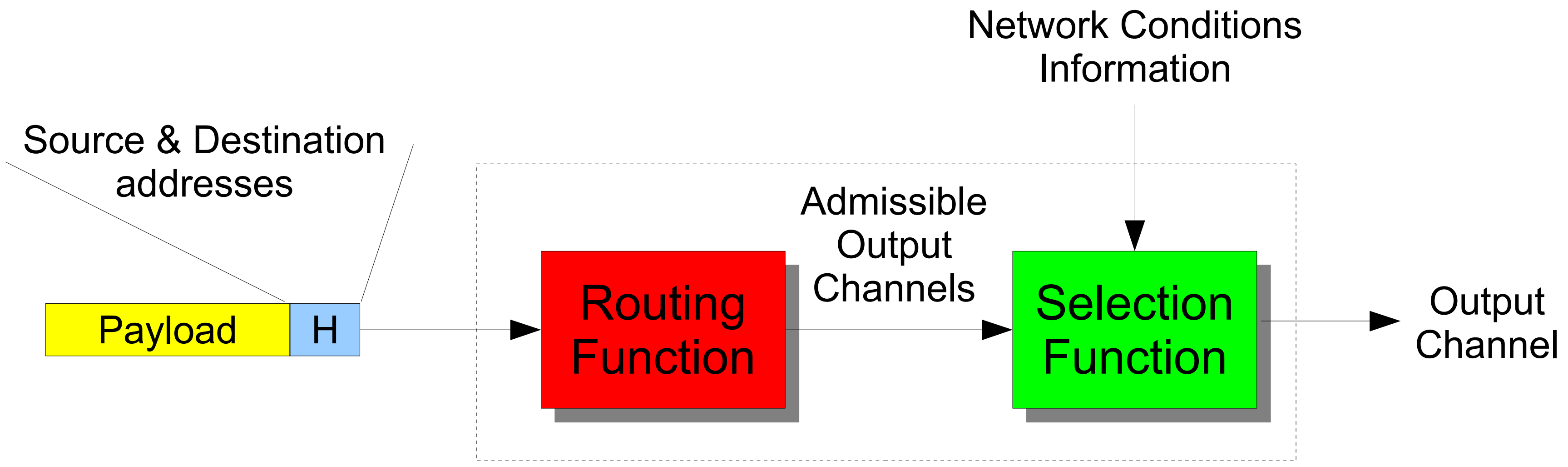
- Use information about the state of the network to make routing decisions
  - Length of queues
  - Historical channel load

Minimal and Non-Minimal

- *Minimal algorithms* only consider minimal routes (shortest path)
- *Non-Minimal algorithms* allow even nonminimal routes

Routing and Selection

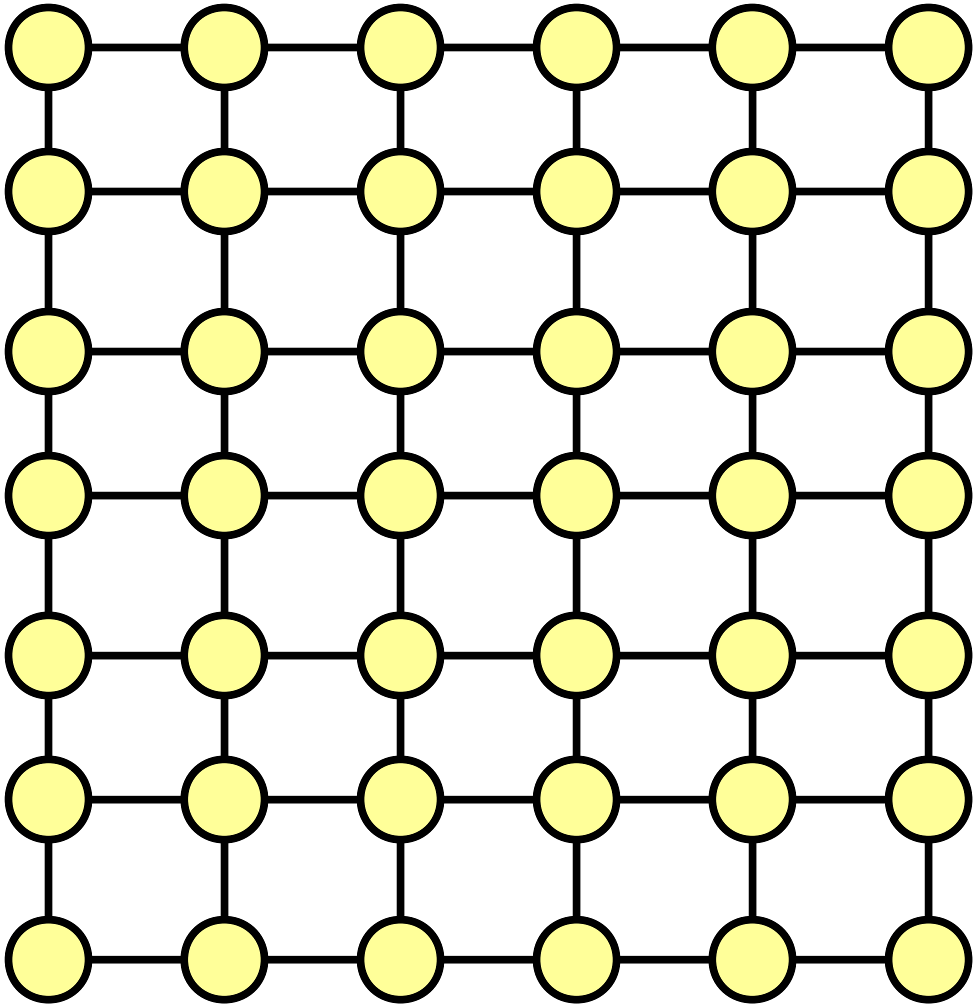
- The routing algorithm can be represented as a *routing relation R selection function S*
- *R* returns a set of paths or channels and *S* selects between the route to be taken


Network Conditions Information
Source & Destination addresses
Admissible Output Channels
Payload
H
Routing Function
Selection Function
Output Channel

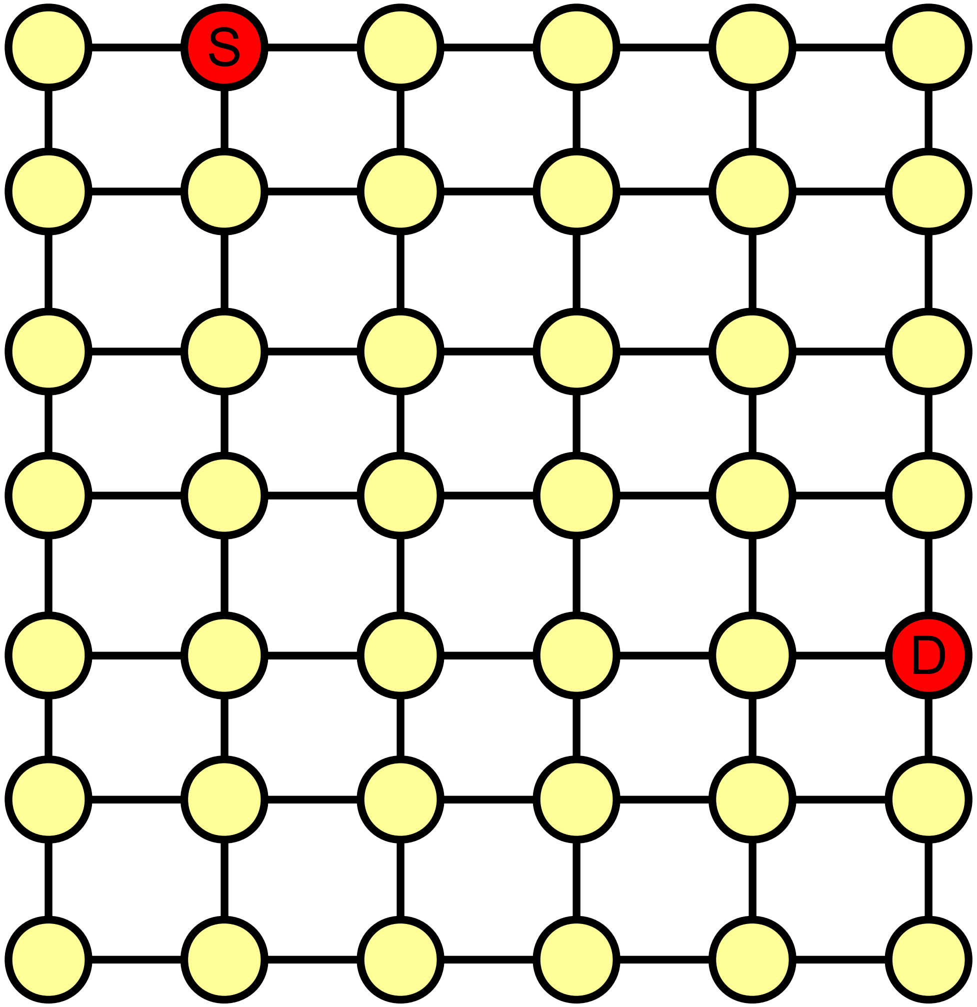
Deterministic Routing

- A packet from a source node *x* to a destination node *y* is always sent over exactly the same route
- Advantages
  - Simple and inexpensive to implement
  - Usual deterministic routing is minimal, which leads to short path length
  - Packets arrive in order
- Disadvantage
  - Lack of path diversity can create large load imbalances

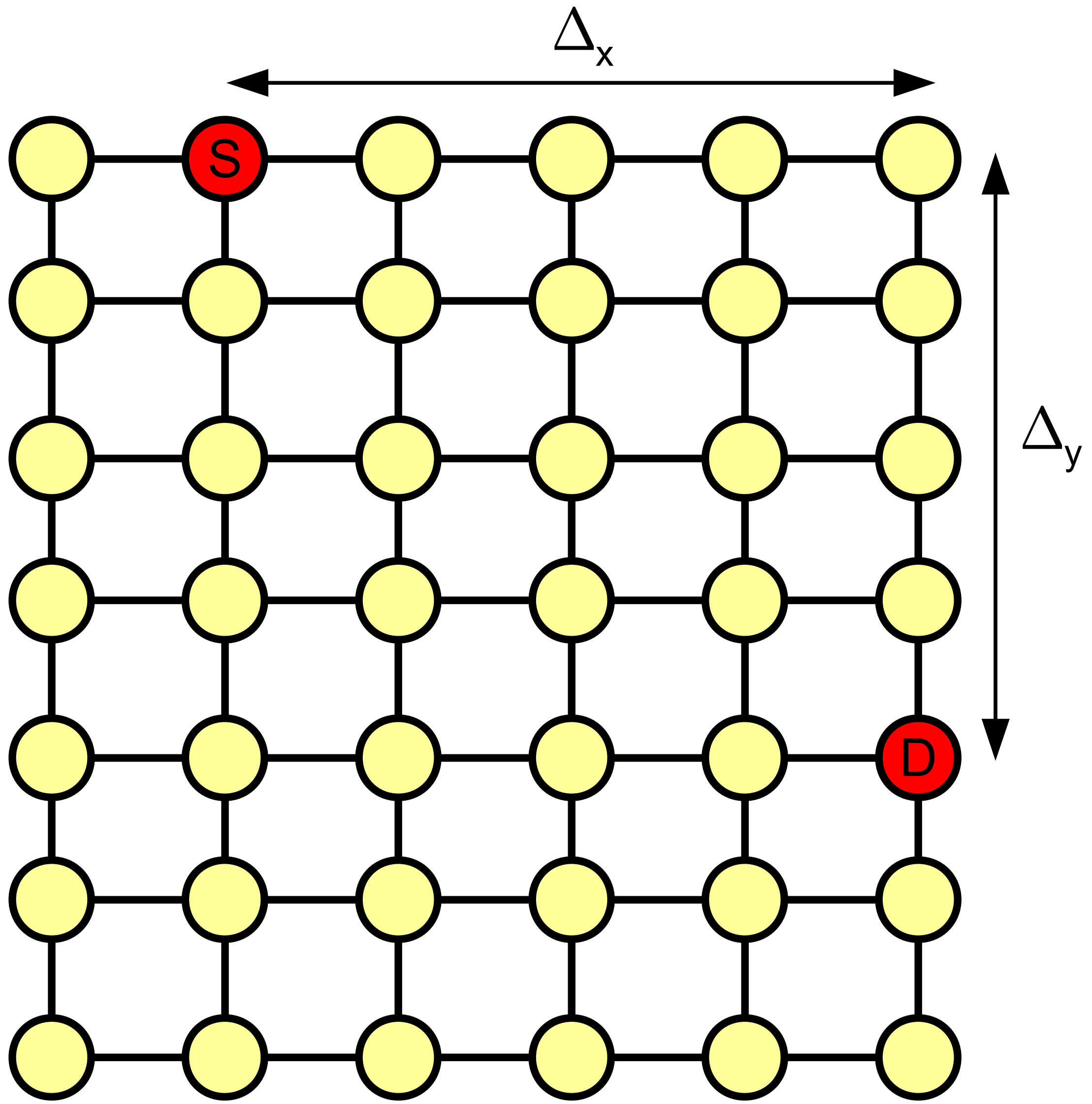
Deterministic XY Routing



Deterministic XY Routing



Deterministic XY Routing



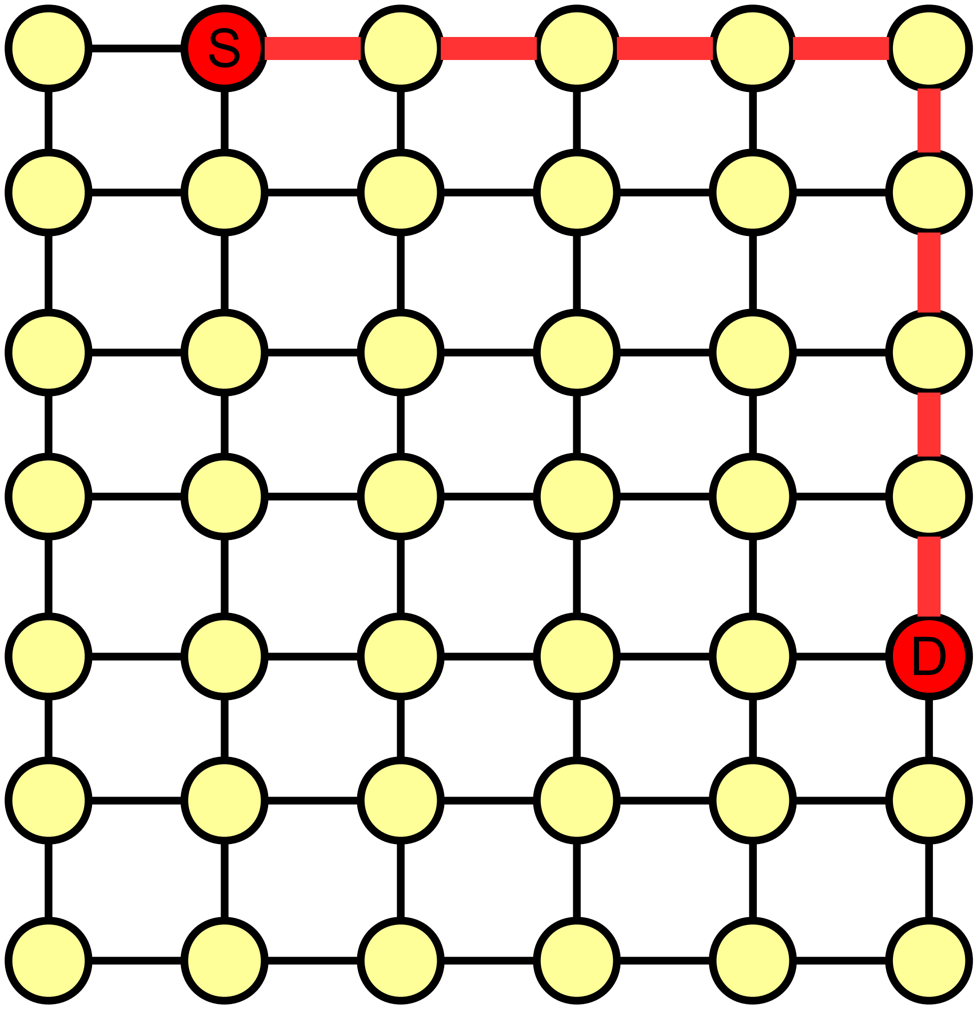
$$\#\text{Paths} = \frac{(\Delta_x + \Delta_y)!}{\Delta_x!\ \Delta_y!}$$

where

$\Delta_x = |S_x - D_x|$

$\Delta_y = |S_y - D_y|$

Deterministic XY Routing



Deadlock

Deadlock (When?)

- When a group of packets cannot make progress, because they are waiting on each other to release resource (buffers, channels)
  - If a sequence of waiting agents form a cycle the network is deadlocked

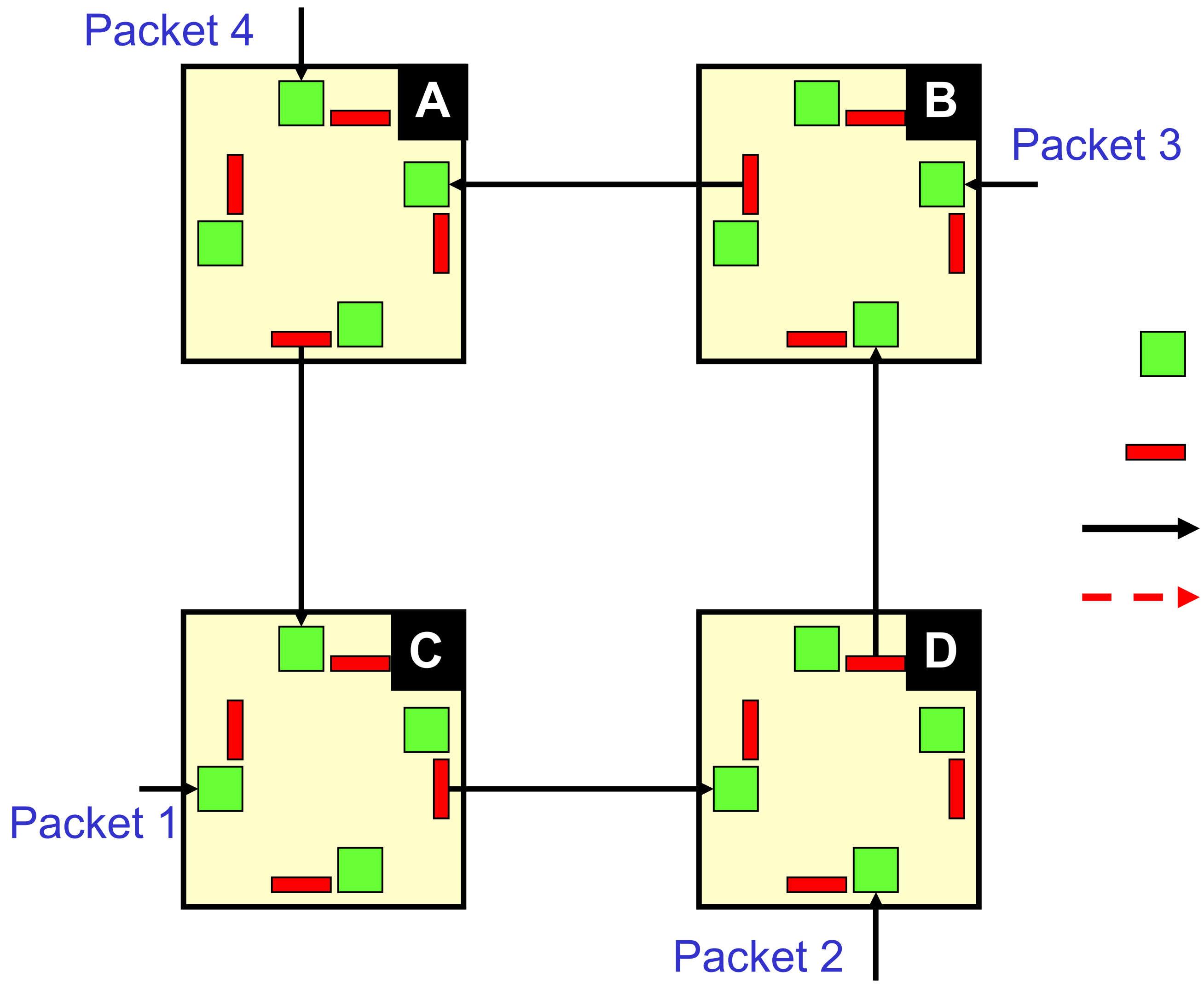
Deadlock (Why?)

- Deadlock can occur if packets are allowed to hold some resources while requesting others

Switching Technique	**Resource**
Store and Forward	Buffer
Virtual Cut-Through	Buffer
Wormhole	Channel

- Because blocked packets holding channels (and their corrisponding flit buffers) remain in the network
  - Wormhole routing is particularly susceptible to deadlock

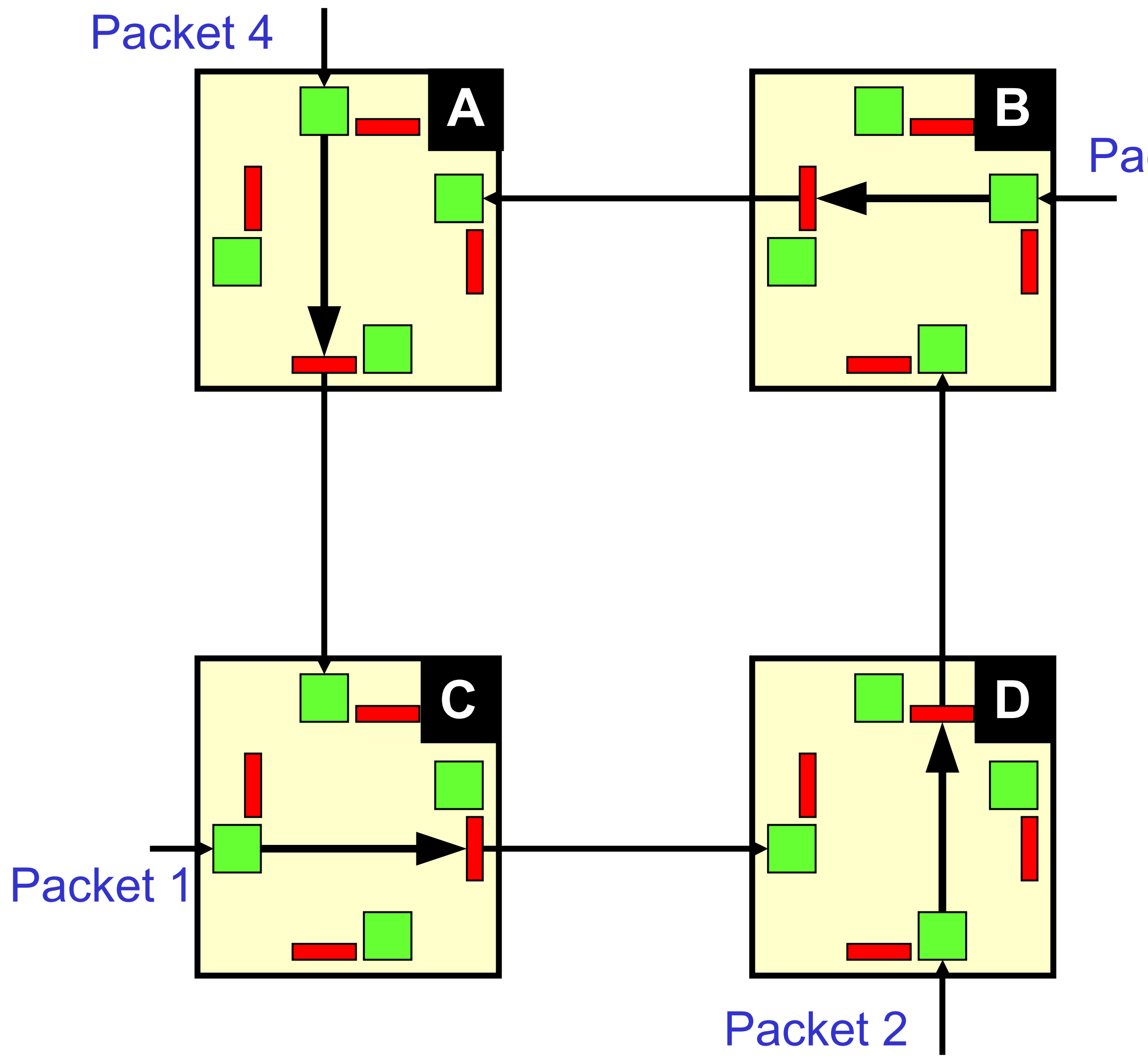
Wormhole Routing Deadlock Example



Packet 1 → B
Packet 2 → A
Packet 3 → C
Packet 4 → D

🟩	Flit buffer
🟥	Input selection circuit
→	Packet progression
⇢	Packet awaiting resource

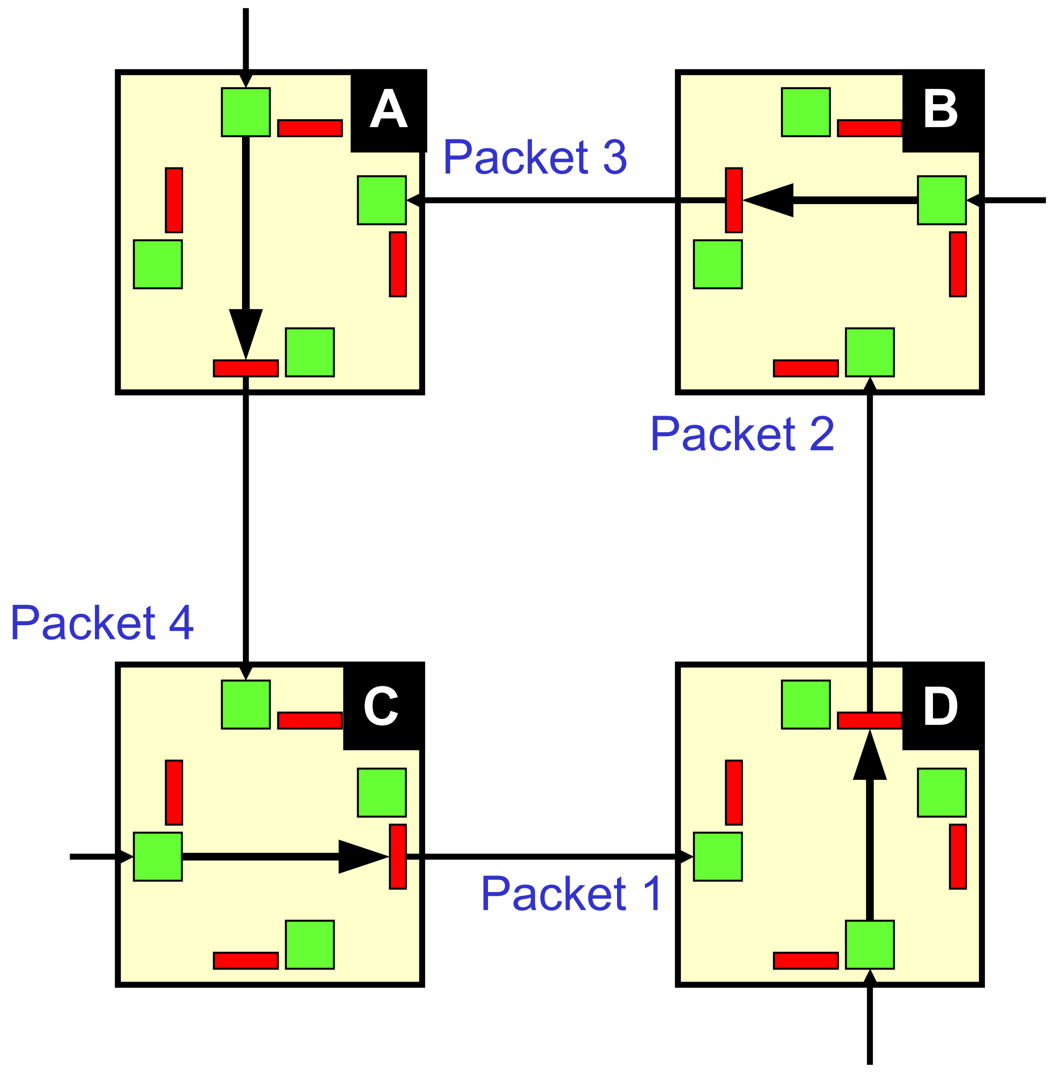
Wormhole Routing Deadlock Example



Packet 1 → B
Packet 2 → A
Packet 3 → C
Packet 4 → D

🟩	Flit buffer
🟥	Input selection circuit
→	Packet progression
⇢	Packet awaiting resource

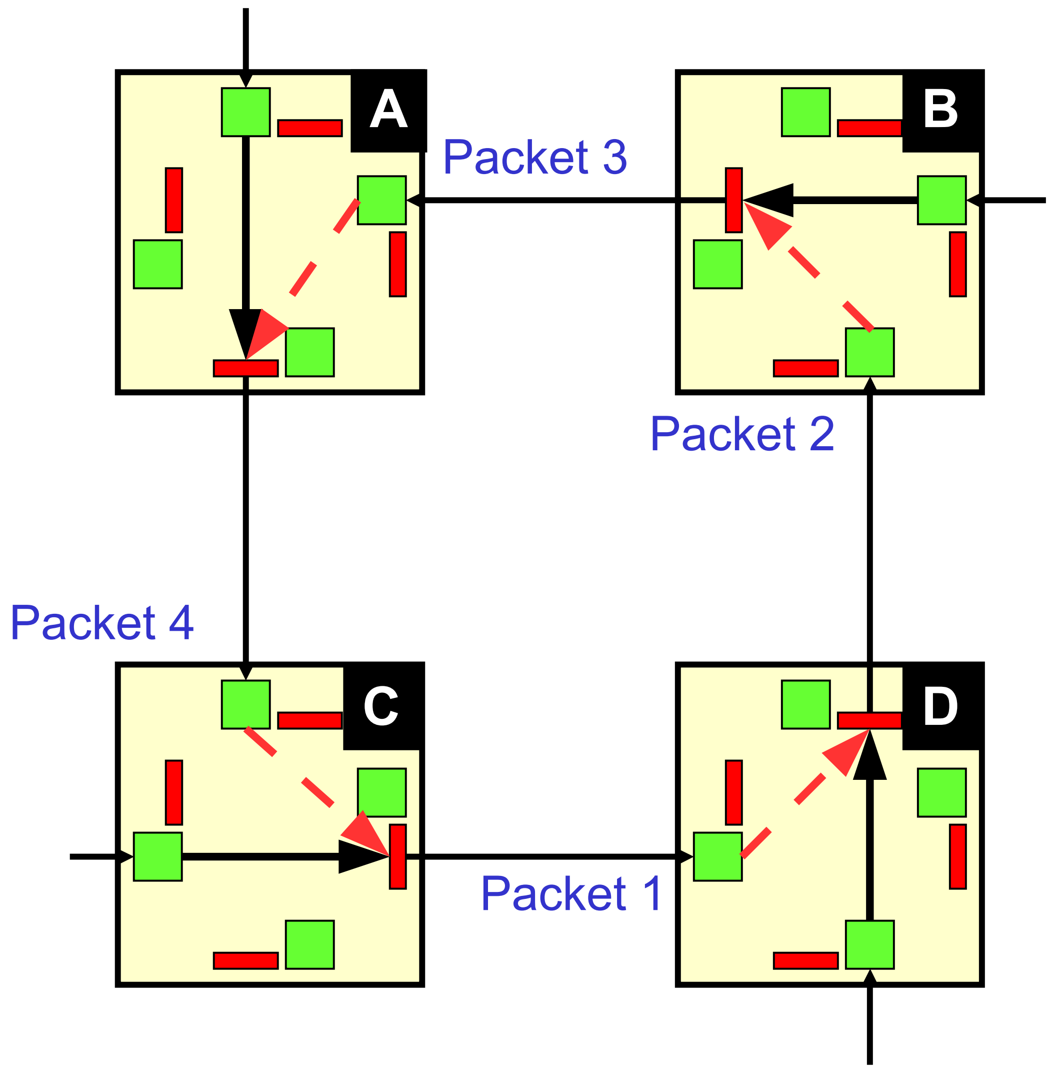
Wormhole Routing Deadlock Example



Packet 1 → B
Packet 2 → A
Packet 3 → C
Packet 4 → D

🟩	Flit buffer
🟥	Input selection circuit
⟶	Packet progression
⇢	Packet awaiting resource

Wormhole Routing Deadlock Example



Packet 1 → B
Packet 2 → A
Packet 3 → C
Packet 4 → D

🟩 Flit buffer

🟥 Input selection circuit

→ Packet progression

- - ▶ Packet awaiting resource

How to Solve Deadlock?

- Allow the preemption of packets involved in a potential deadlock situation
- Preemption packets can be
  - Rerouted
    - Adaptive nonminimal routing techniques
  - Discarded
    - Packets recovered at the source and restransmitted
- Not used in most direct networks architectures
  - Requirements of low-latency and reliability

How to Solve Deadlock?

- More commonly, deadlock is avoided by the routing algorithm
  - By ordering network resources and requiring that packets use these resources in strictly monotonic order
    - Circular wait is avoided

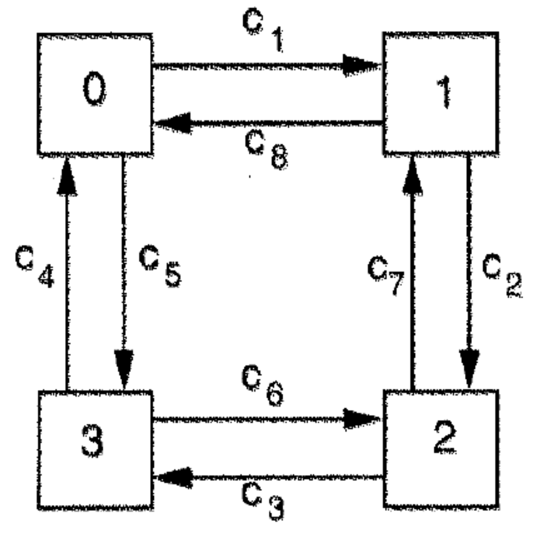
Channel Dependecy Graph

- In wormhole routing networks channels are the critical resources
- There is a *Direct Dependency* from l_i to l_j if l_j can be used immediately after l_i by messages destined to some node n
- The *Channel Dependency Graph* for a network and a routing algorithm is a direct graph $D=G(L,D)$
  - L consists of all the unidirectional channels in the network
  - D includes the pairs (l_i, l_j) if there is a direct dependency from l_i to l_j

Duato’s Theorem

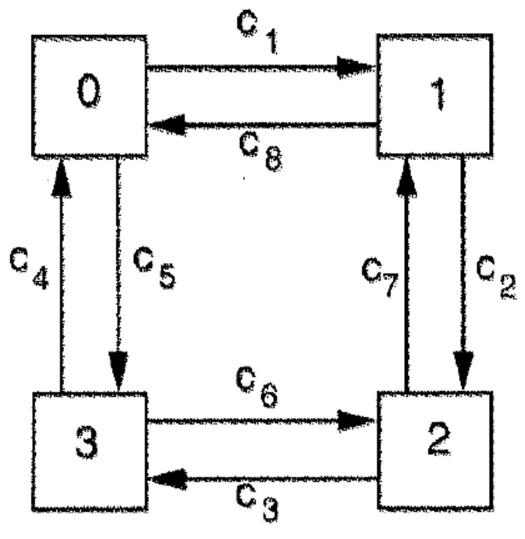
- A routing function R is *deadlock-free* if there are no cycles in its channel dependency graph

CDG Example



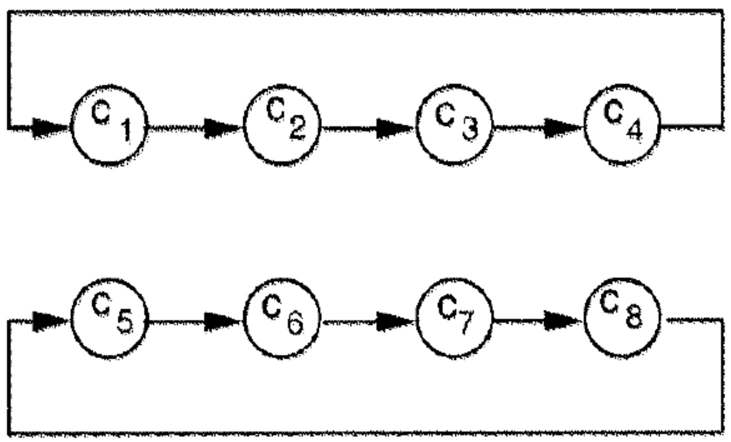
(a)

CDG Example



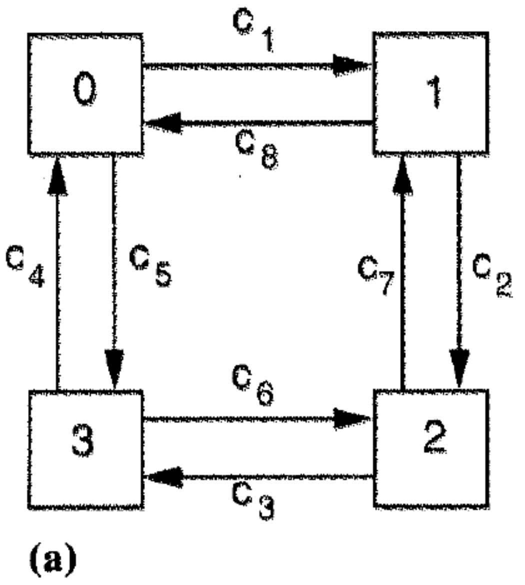
(a)

CDG



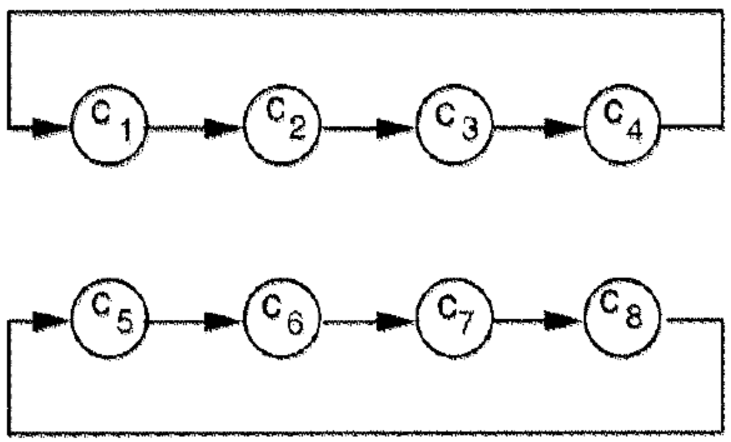
(b)

CDG Example



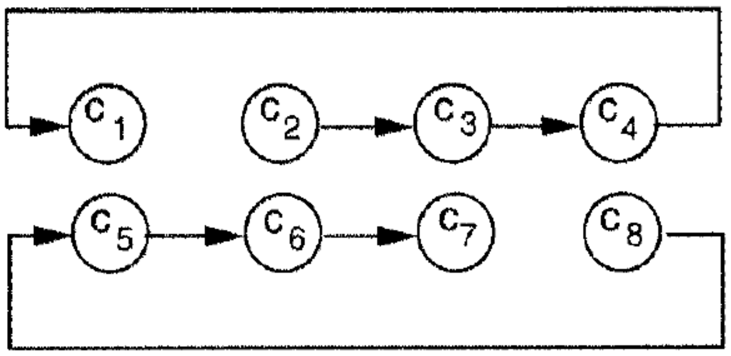
(a)

CDG



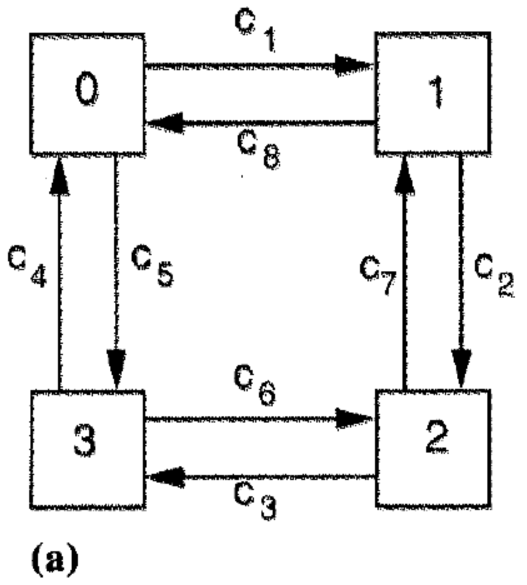
(b)

CDG without cycles

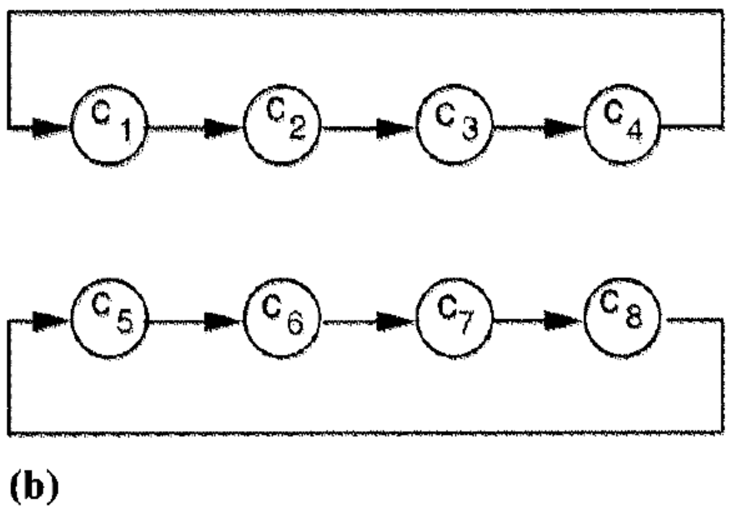


(c)

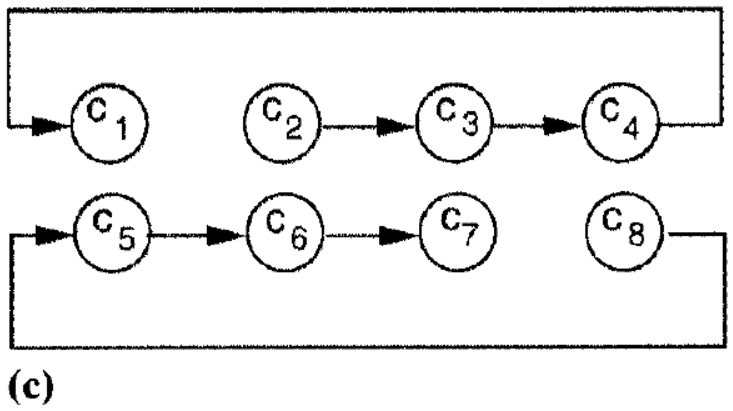
CDG Example



CDG



CDG without cycles



- The routing is still minimal
  - ➔ However, to send a packet from 0 to 2, the packet must be forwarded through node 3

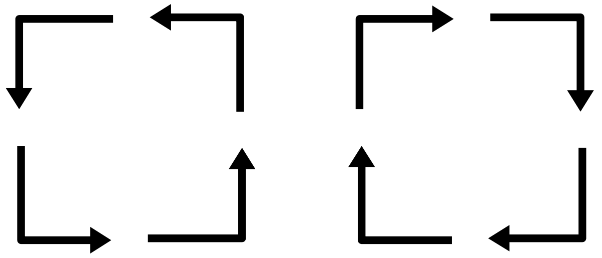
The Turn Model

- The *Turn Model* provide a systematic approach to the development of maximally adaptive routing algorithms

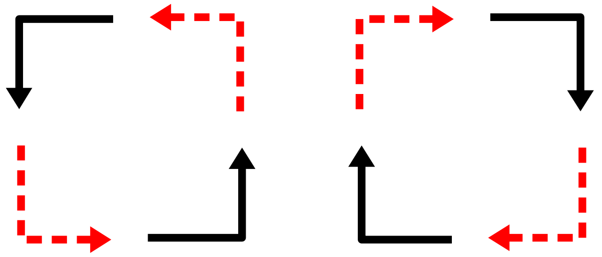
The Turn Model

- The *Turn Model* provide a systematic approach to the development of maximally adaptive routing algorithms
    1. Classify channels according to the direction in which they route packets
    2. Identify the turns that occur between one direction and another
    3. Identify the simple cycles these turns can forms
    4. Prohibit one turn in each cycle

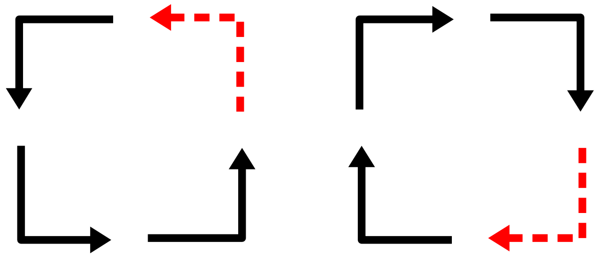
The Turn Model



Abstract cycles in a 2D mesh

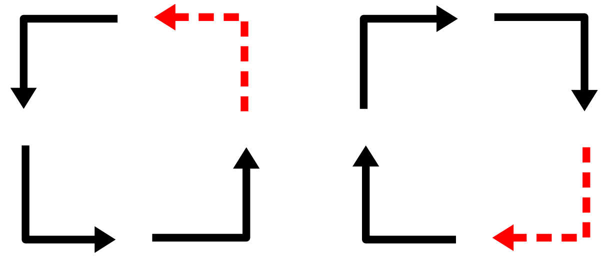


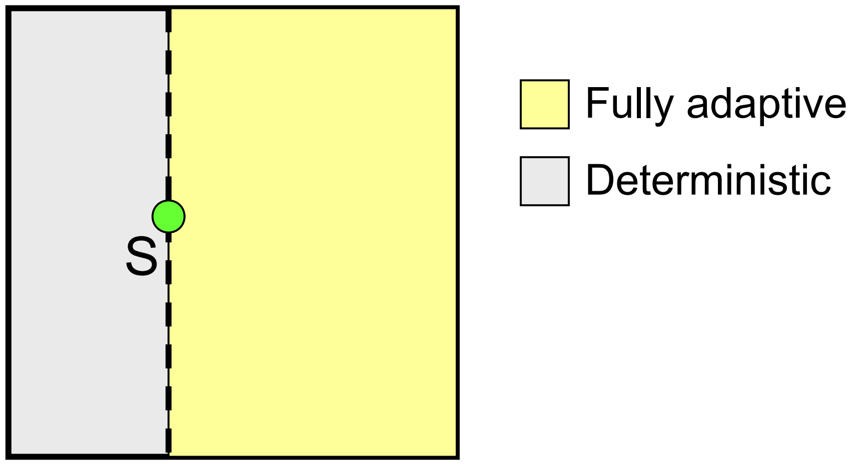
Four turns allowed in XY routing



Six turns allowed in *west-first* routing

West-First Routing

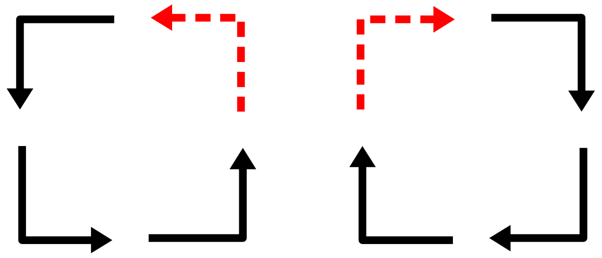


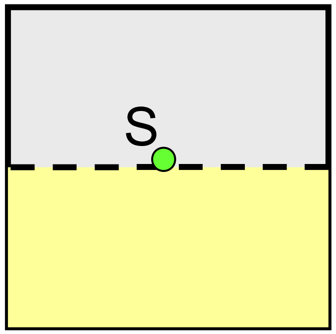


- First route a packet west, if necessary, and then adaptively south, east, and north
  - The algorithm is fully adaptive if the destination is on the right-hand side (east) of the source
    - Otherwise it is deterministic

Routing Algorithms based on Turn Model

North-last

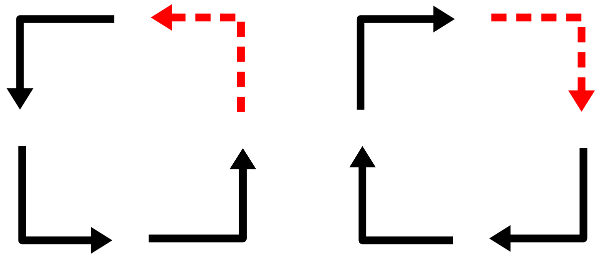


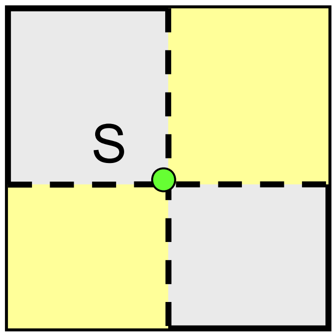


🟨 Fully adaptive

⬜ Deterministic

Negative-first





🟨 Fully adaptive

⬜ Deterministic

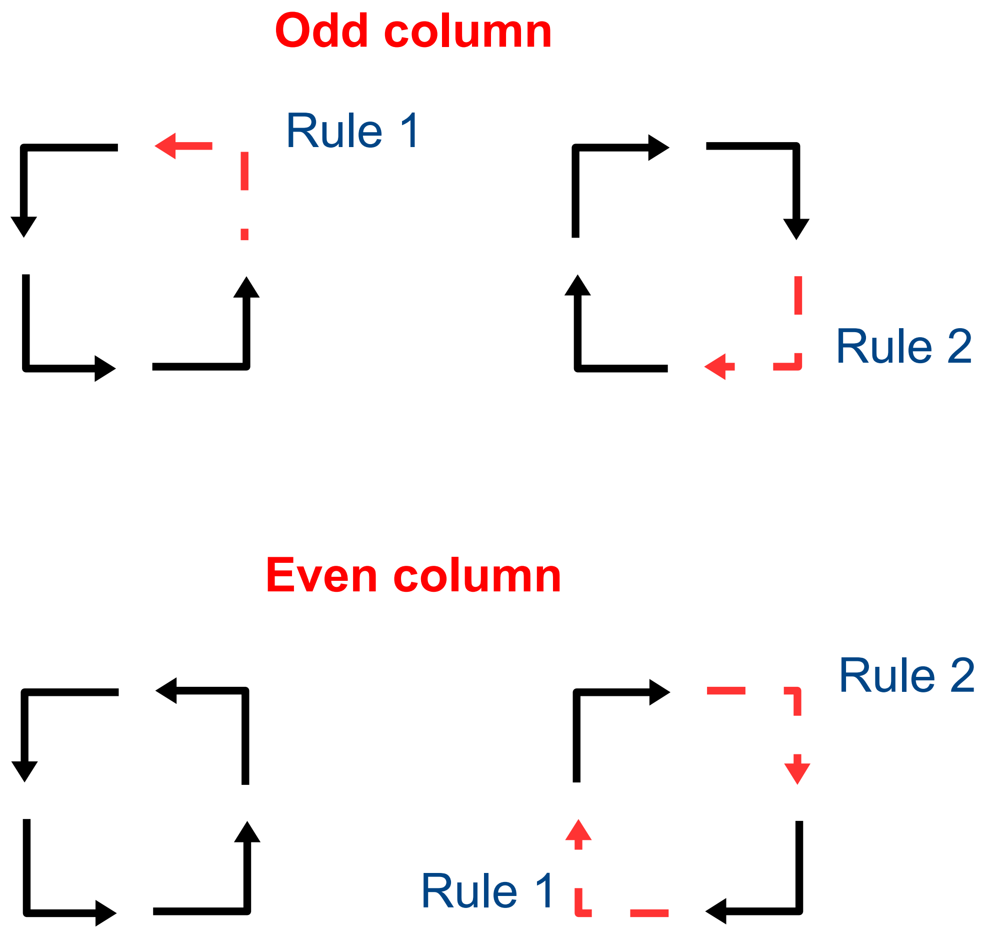
Load Balancing

- Turn model derived routing algorithms do not uniformly distribute the traffic over the network
  - For a given source destination pair
    - A single path
    - All minimal paths

Odd-Even Routing

- Governed by two rules
  - Rule 1. Any packet is *not* allowed to take an EN turn at any nodes located in an even column, and it is *not* allowed to take an NW turn at any nodes located in an odd column
  - Rule 2. Any packet is *not* allowed to take an ES turn at any nodes located in an even column, and it is *not* allowed to take an SW turn at any nodes located in an odd column

Odd-Even Routing



Degree of Adaptiveness

$h = d_x/2$

$h' = (d_x-1)/2$

Destination on the right (Δx >0)

$$P_{\text{odd−even turn model}} = \begin{cases} \frac{(d_y+h')!}{d_y!h'!} & \text{if column } x_s \text{ is an allowable column and } d_x \text{ is an odd number,} \\ \frac{(d_y+h)!}{d_y!h!} & \text{otherwise.} \end{cases}$$

Destination on the left (Δx <0)

$$P_{\text{odd−even turn model}} = \begin{cases} \frac{(d_y+h)!}{d_y!h!} & \text{if column } x_s \text{ is an allowable column or } \Delta x = 0, \\ \frac{(d_y+h')!}{d_y!h'!} & \text{otherwise.} \end{cases}$$

Routing in Reconfigurable Networks

- Routing algorithms have been developed for each type of network topology
- If the topology changes
  - Routing algorithm has to change
- The router must be flexible and programmable to allow for the implementation of different deadlock-free algorithms
- Two techniques
  - Source routing
  - Table-lookup routing

Routing in Reconfigurable Networks

- Two techniques
  - Source routing
  - Table-lookup routing

Source Routing

- The source node specifies the routing path
- The packet must carry complete routing information in the packet header
  - It is important to minimize packet length
- Street-sign routing
  - [<*Node*$_1$,*Turn*$_1$>,<*Node*$_2$,*Turn*$_2$>,…]
  - By default packets arriving from the input channel +X (+Y) will be forwarded to -X (-Y)
  - If current node is *Node*$_i$ then turn *Turn*$_i$ is taken and <*Node*$_i$,*Turn*$_i$> is removed from the header

Source Routing – Example

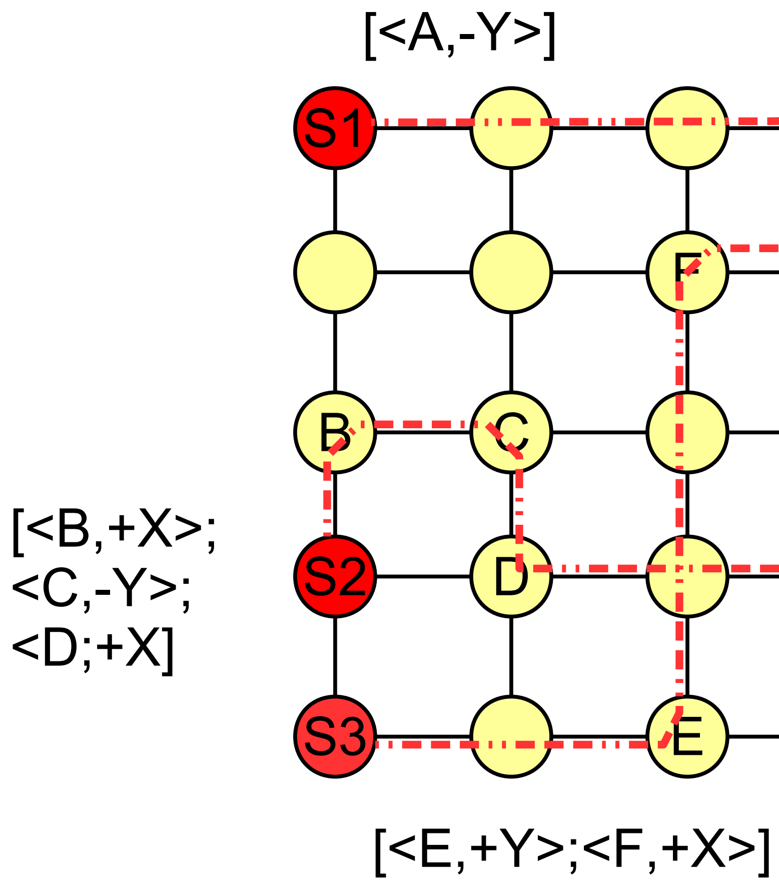
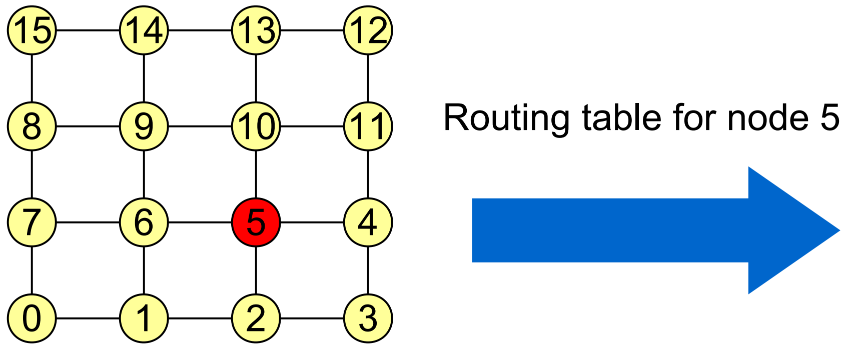


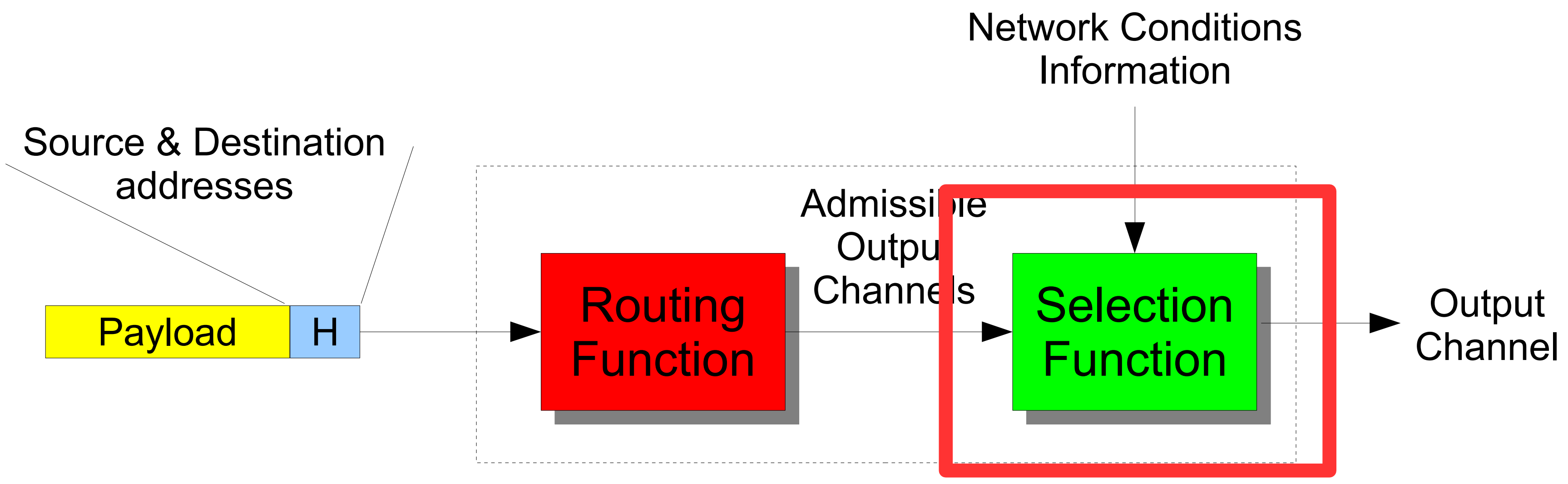
Table-lookup Routing

- Amenable to reconfigurable topologies
- An obvious implementation
  - Given a destination node address the corresponding entry in the table indicates which outgoing channel should be used to forward the packet
  - Not practical
    - Large table is inefficient in the use of chip area



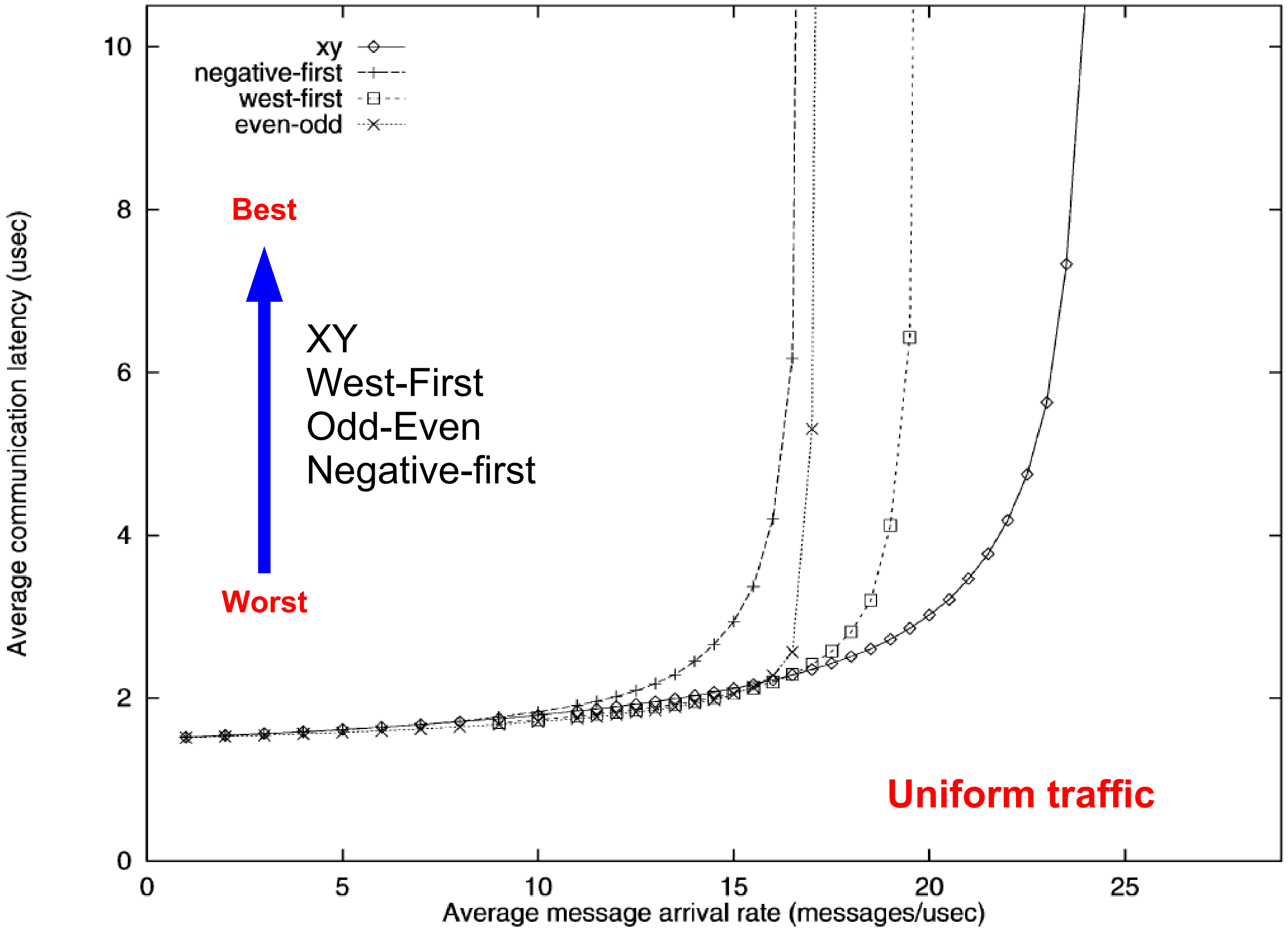
Dst	Out
0	West
1	West
2	South
3	East
4	East
5	-
6	West
7	West
8	West
9	West
10	North
11	East
12	East
13	North
14	West
15	West

Selection Function

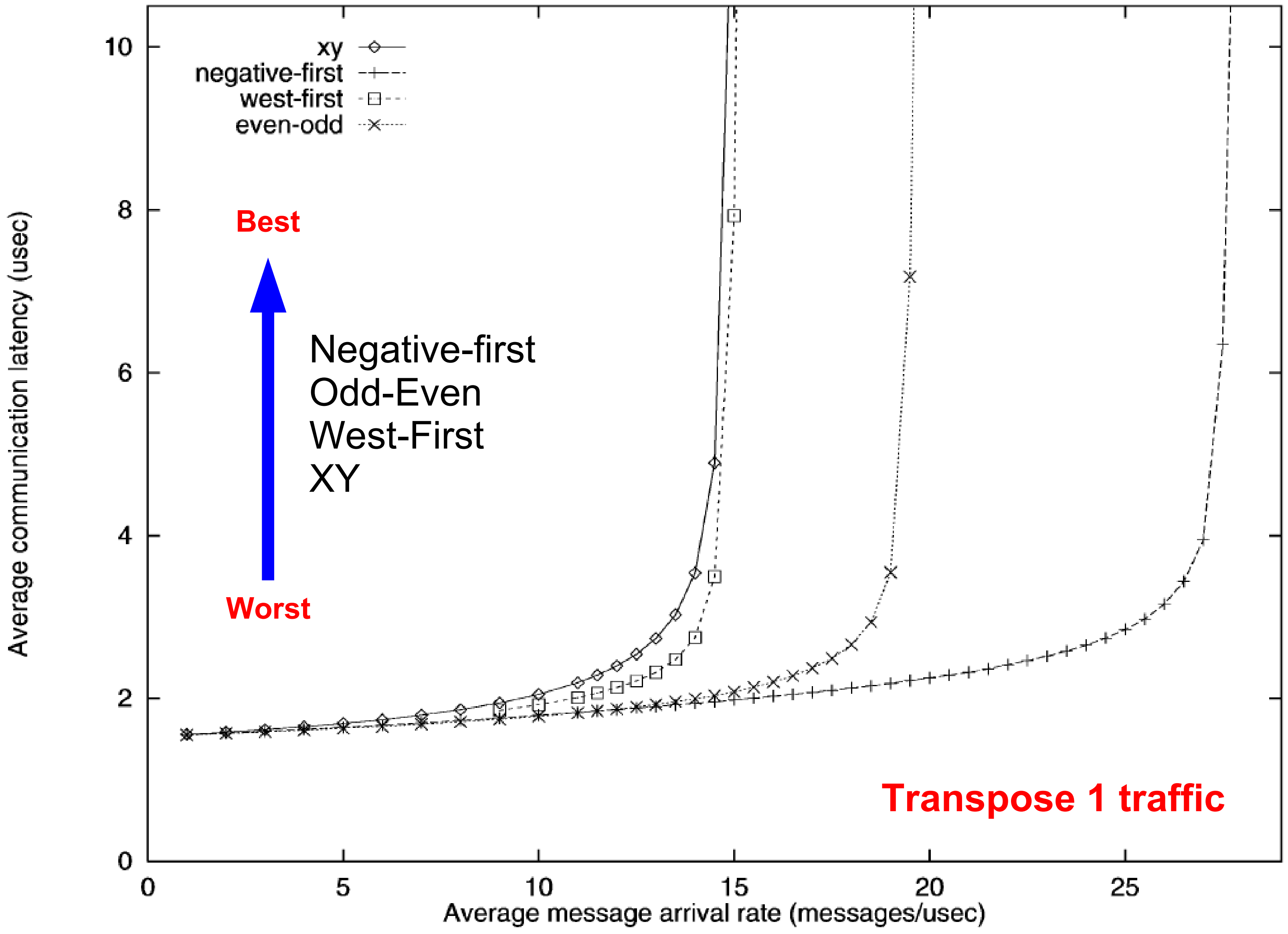

Network Conditions Information
Source & Destination addresses
Admissible Output Channels
Payload
H
Routing Function
Selection Function
Output Channel

- Random
- Buffer level
- Neighbors-on-Path

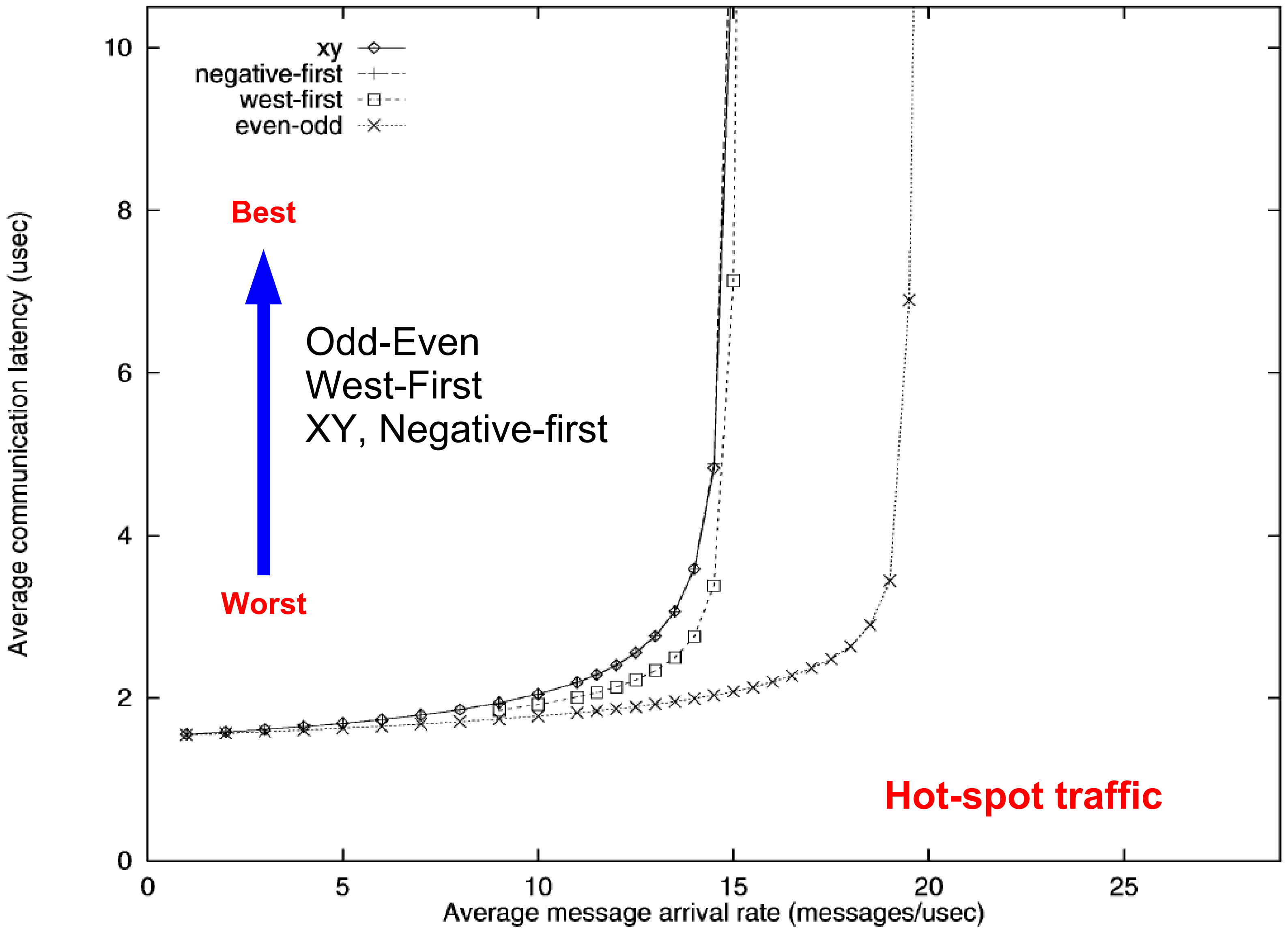
Performance of Routing Algorithms



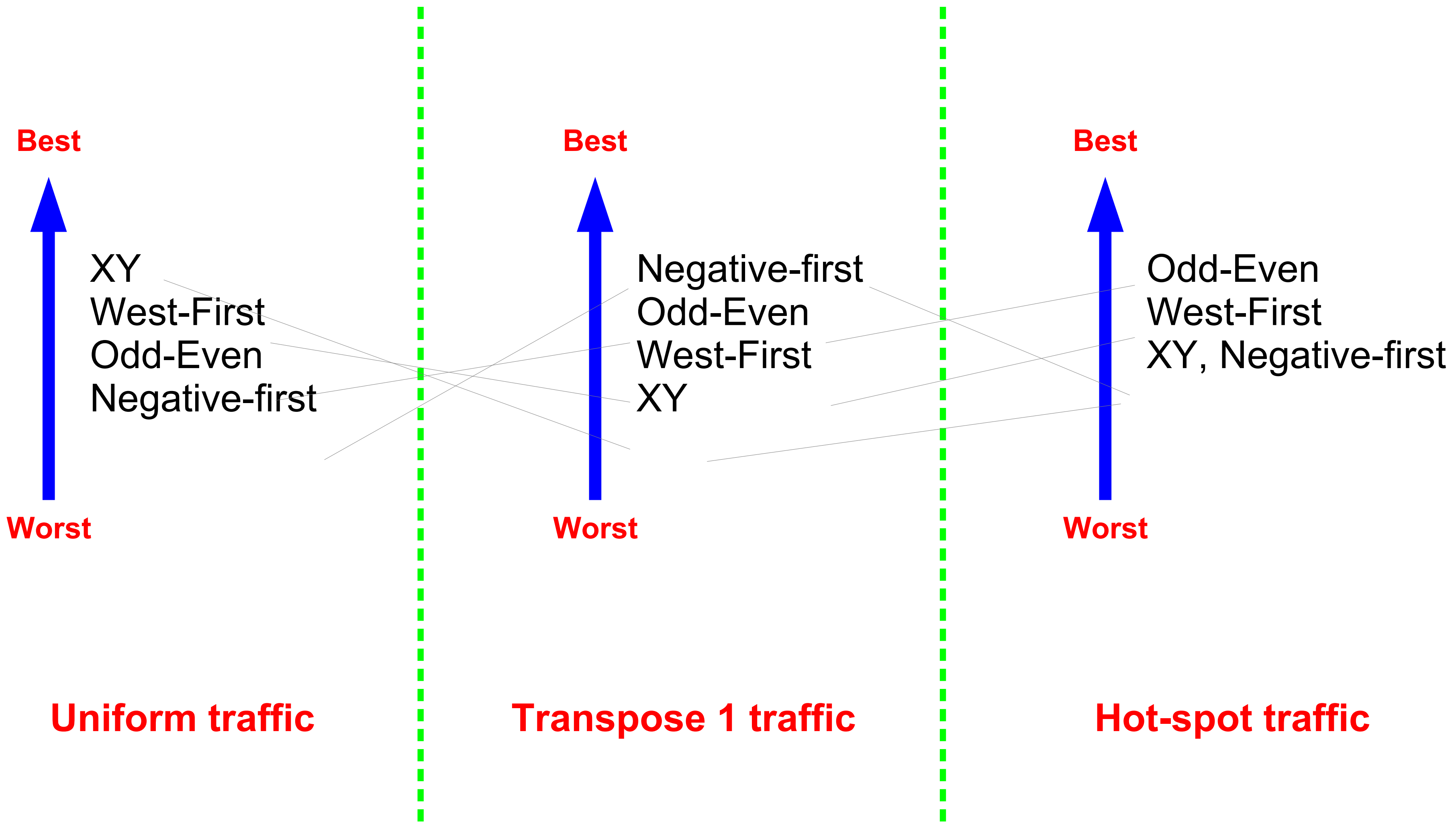
Performance of Routing Algorithms



Performance of Routing Algorithms



No Winner Routing Algorithm



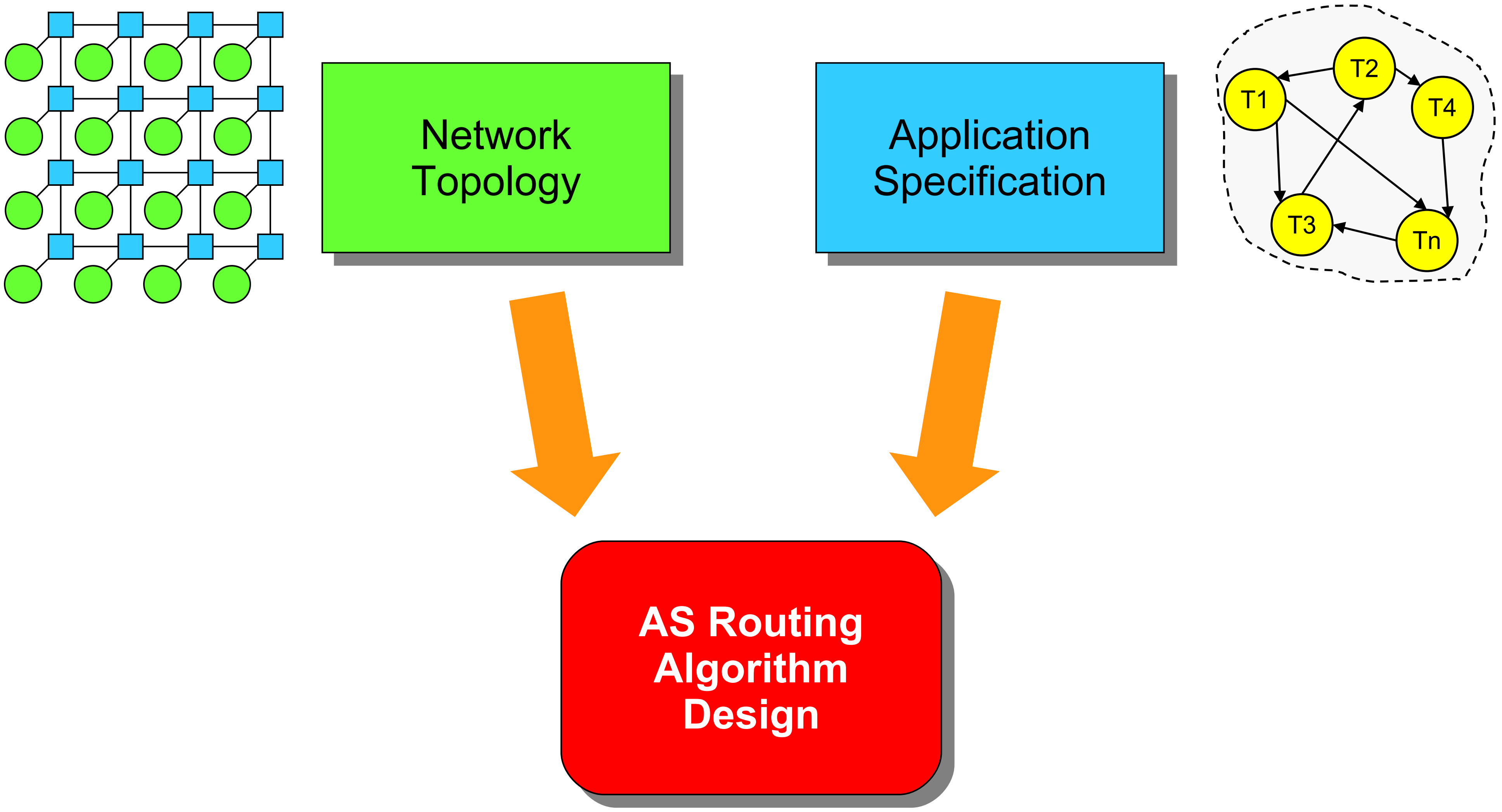
Idea

- Why don't design the routing algorithm based on the specific traffic scenario?

Plenty of Information

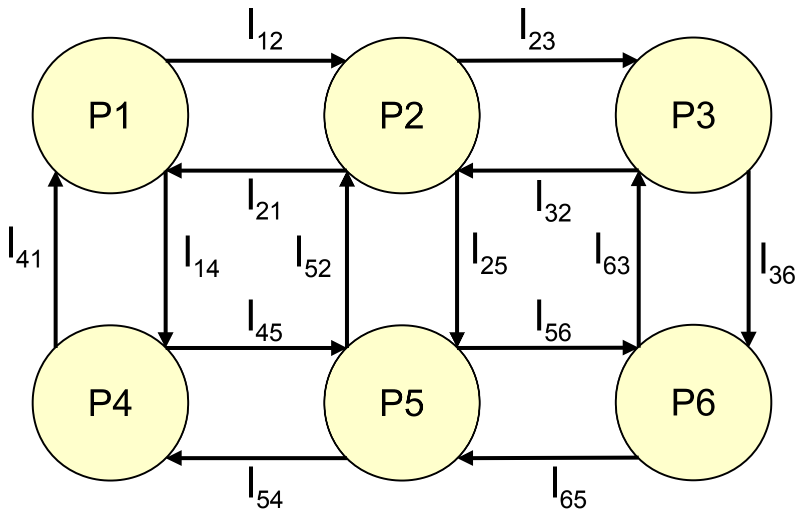
- Tasks which communicate and tasks which do never communicate
- After task mapping
  - Information about network nodes which communicate
- Cuncurrent/non cuncurrent communications
- Communications bandwidth requirements

APSRA Design Flow



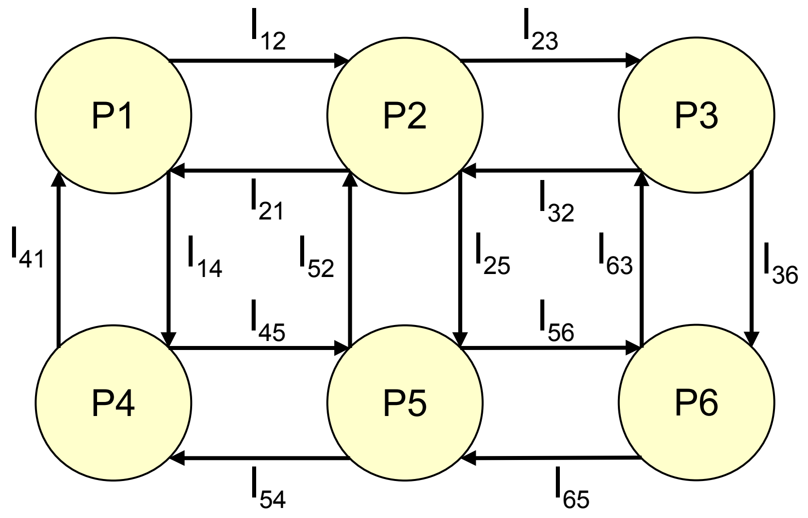
APSRA by Example

Topology Graph

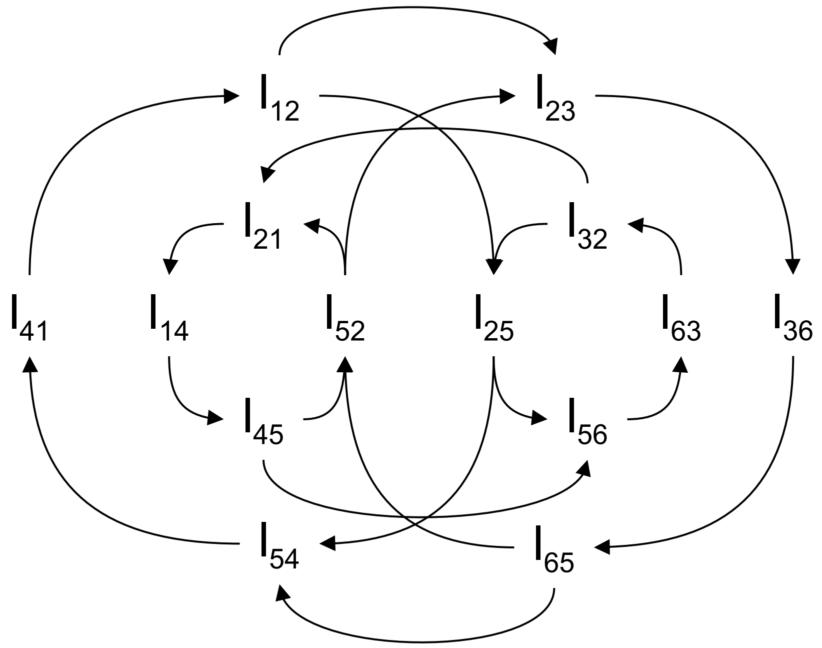


APSRA by Example

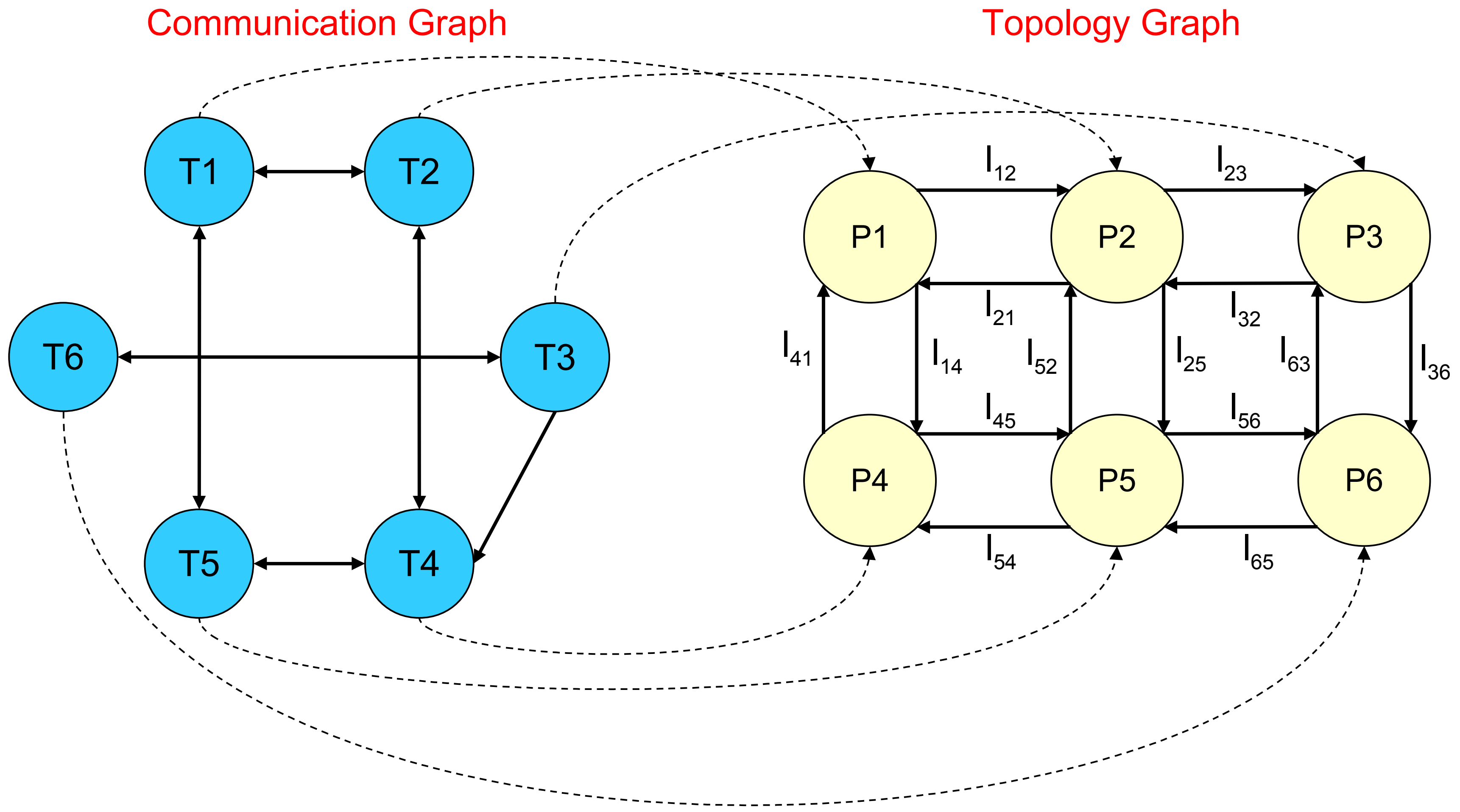
Topology Graph



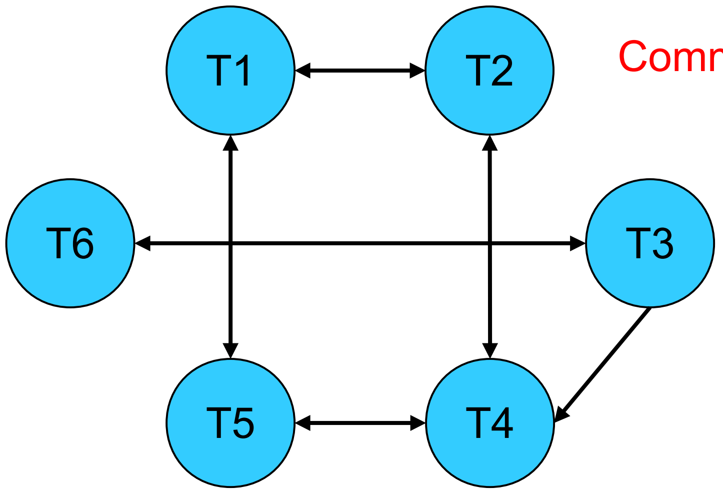
Channel Dependency Graph



APSRA by Example

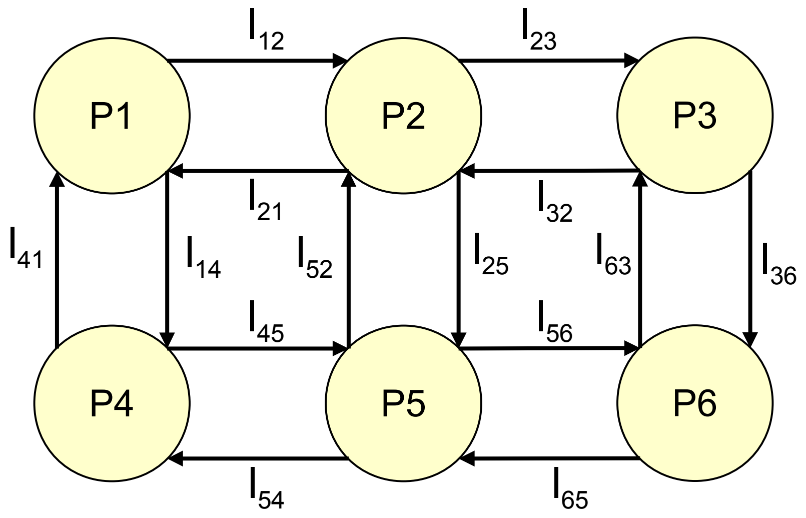


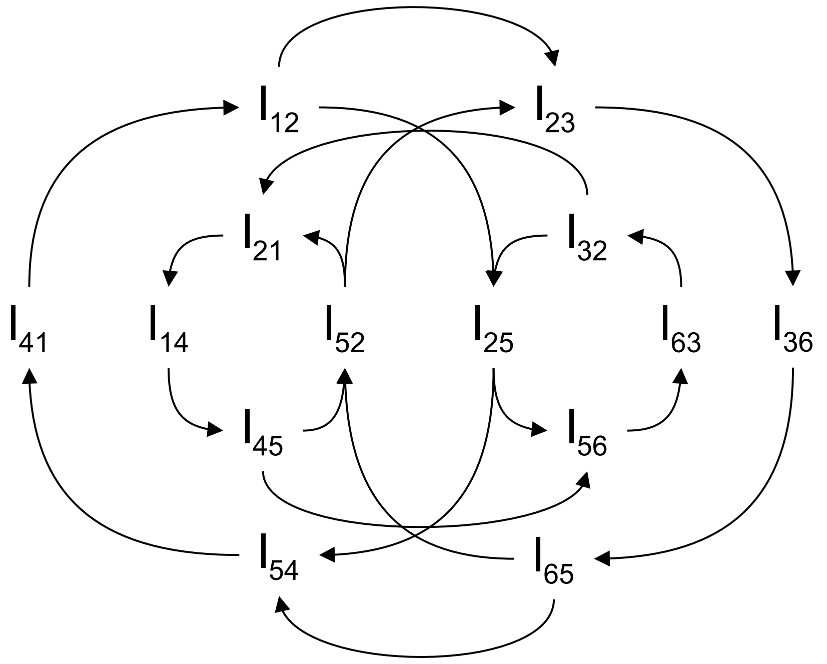
APSRA by Example



Communication Graph

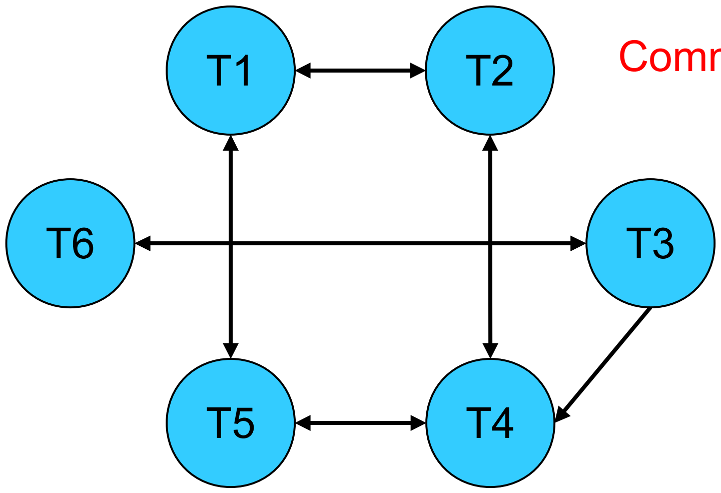
Topology Graph





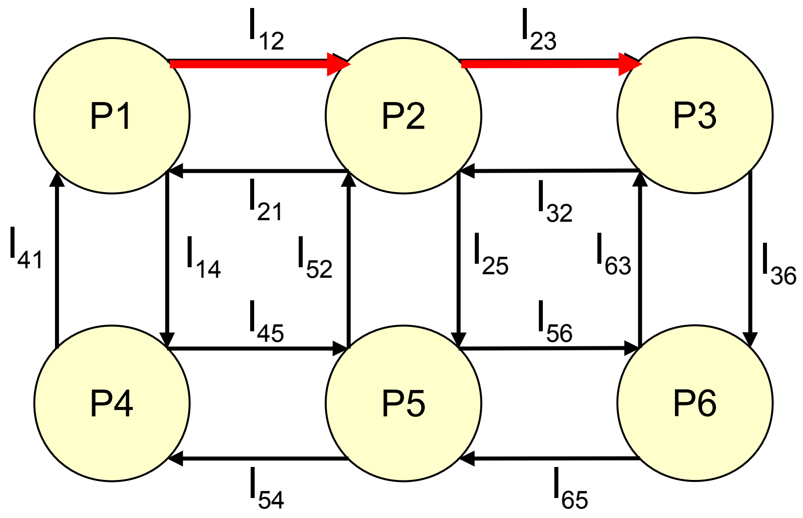
Channel Dependency Graph

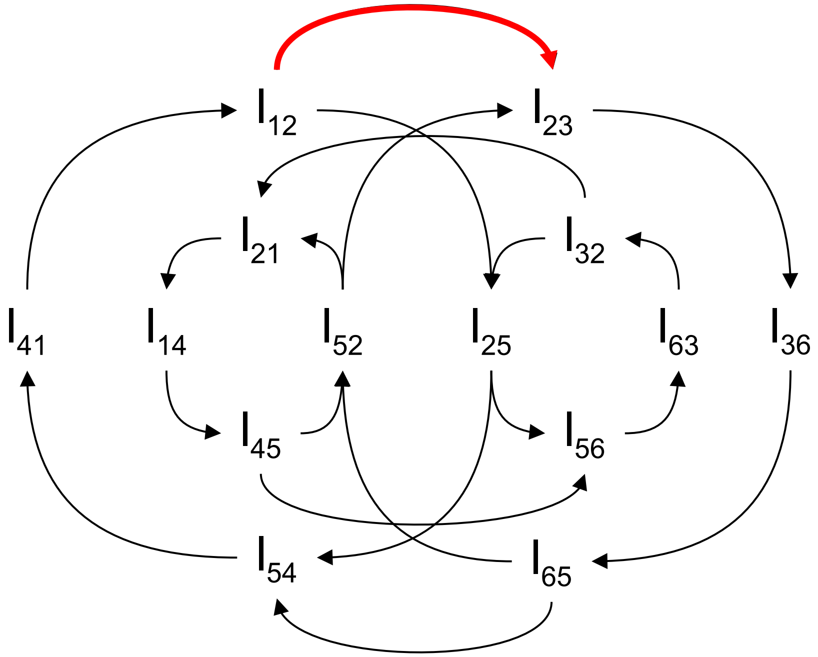
APSRA by Example



Communication Graph

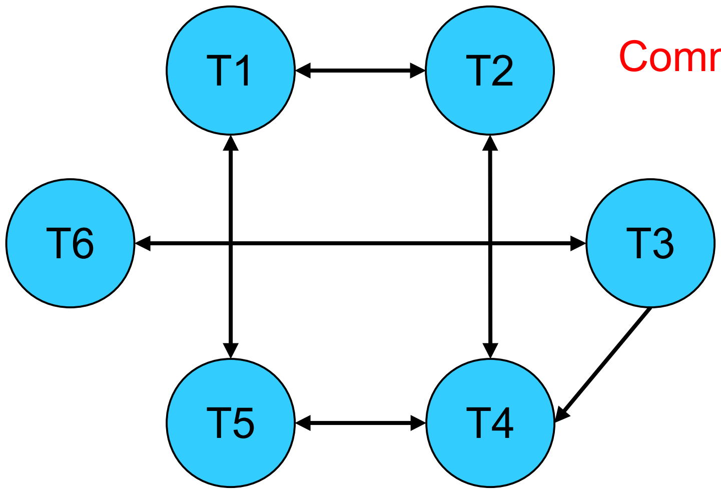
Topology Graph





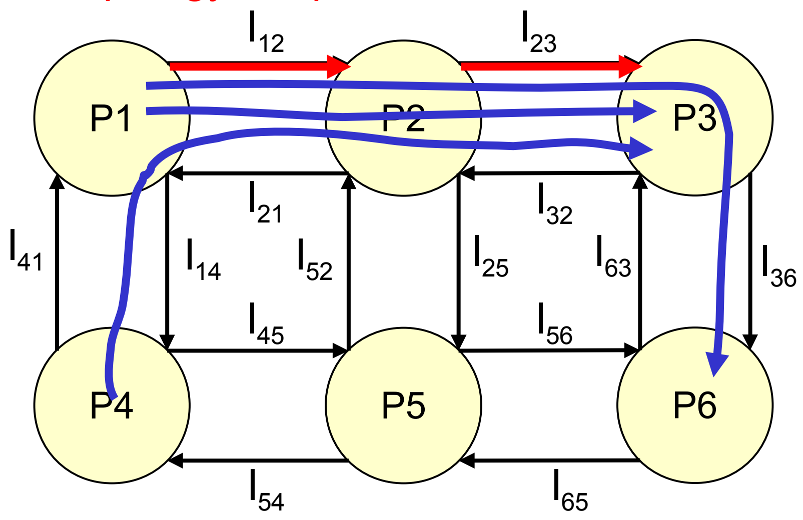
Channel Dependency Graph

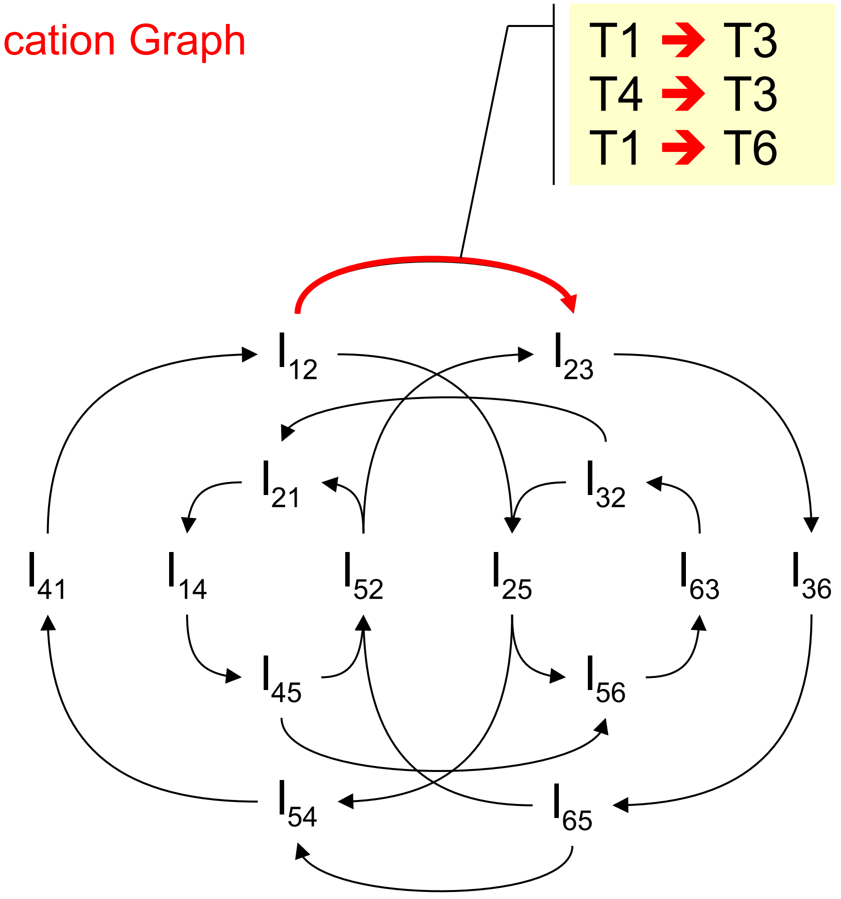
APSRA by Example



Communication Graph

Topology Graph

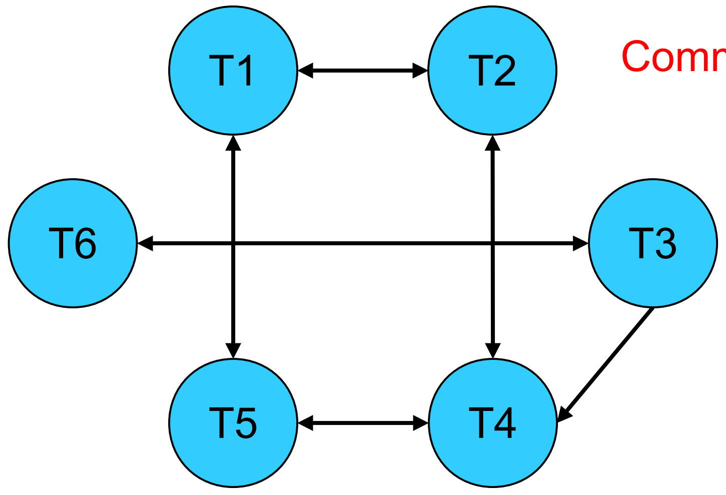




T1 ➔ T3
T4 ➔ T3
T1 ➔ T6

Channel Dependency Graph

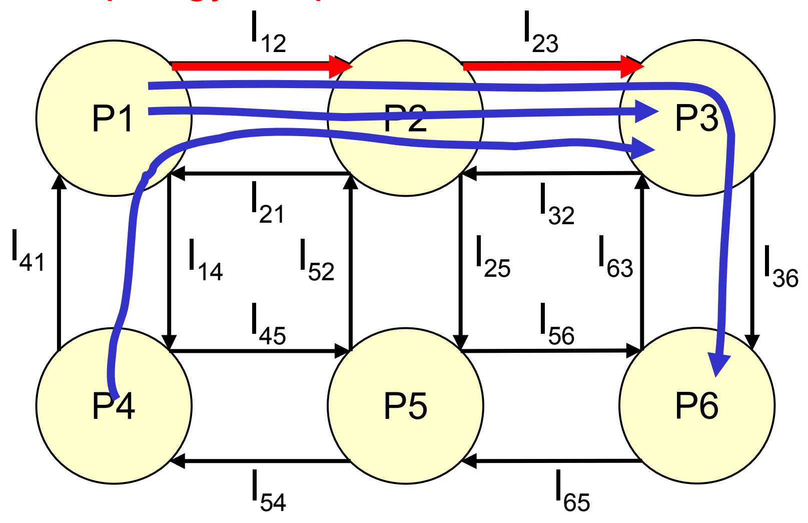
APSRA by Example

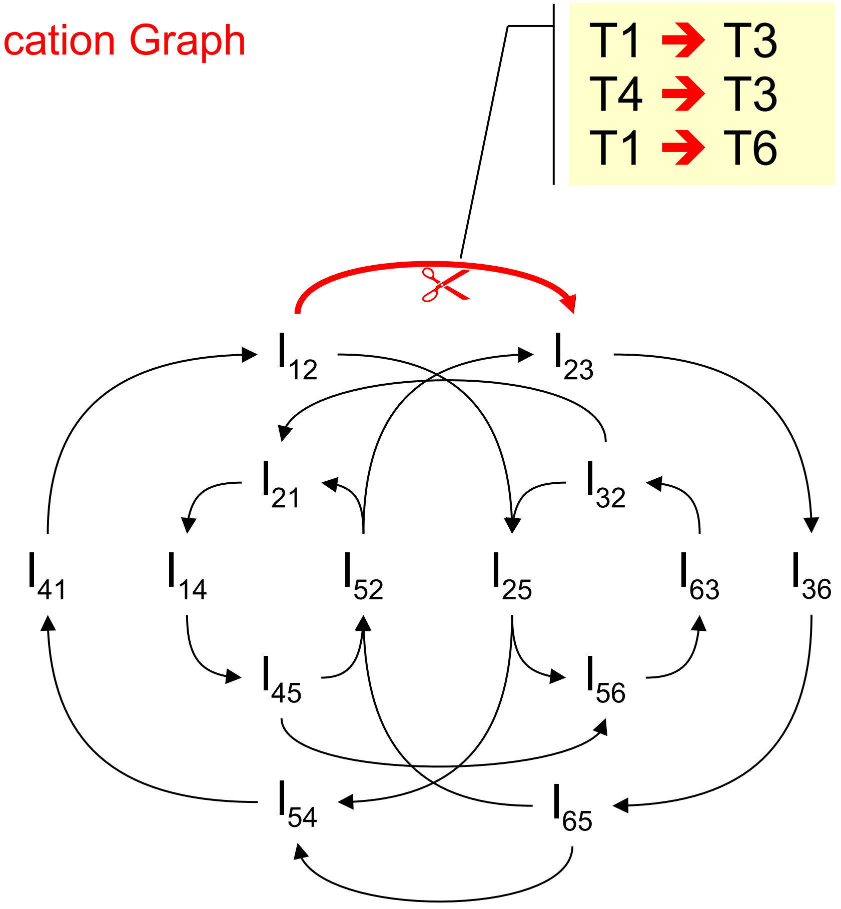


Communication Graph

T1 ➔ T3
T4 ➔ T3
T1 ➔ T6

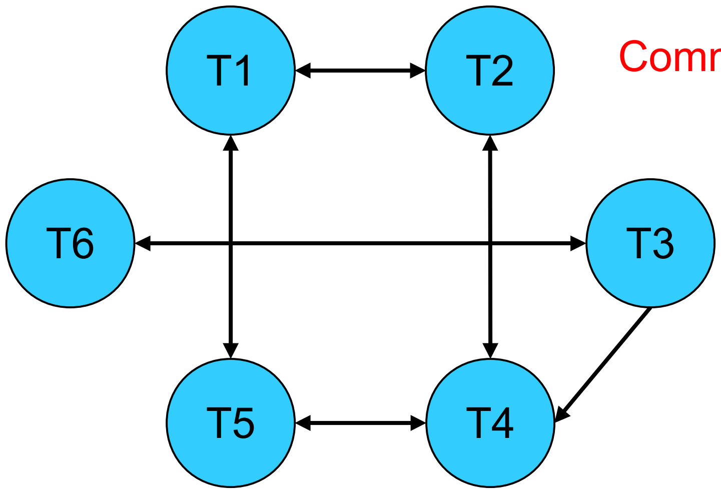
Topology Graph





Channel Dependency Graph

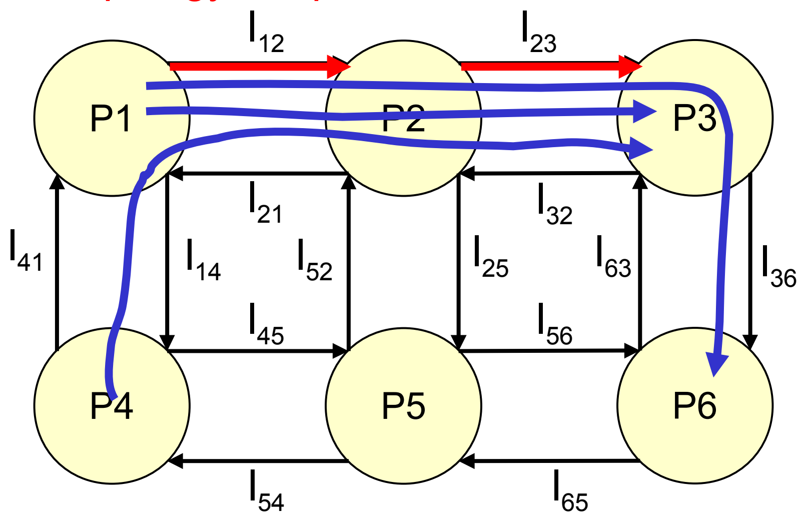
APSRA by Example

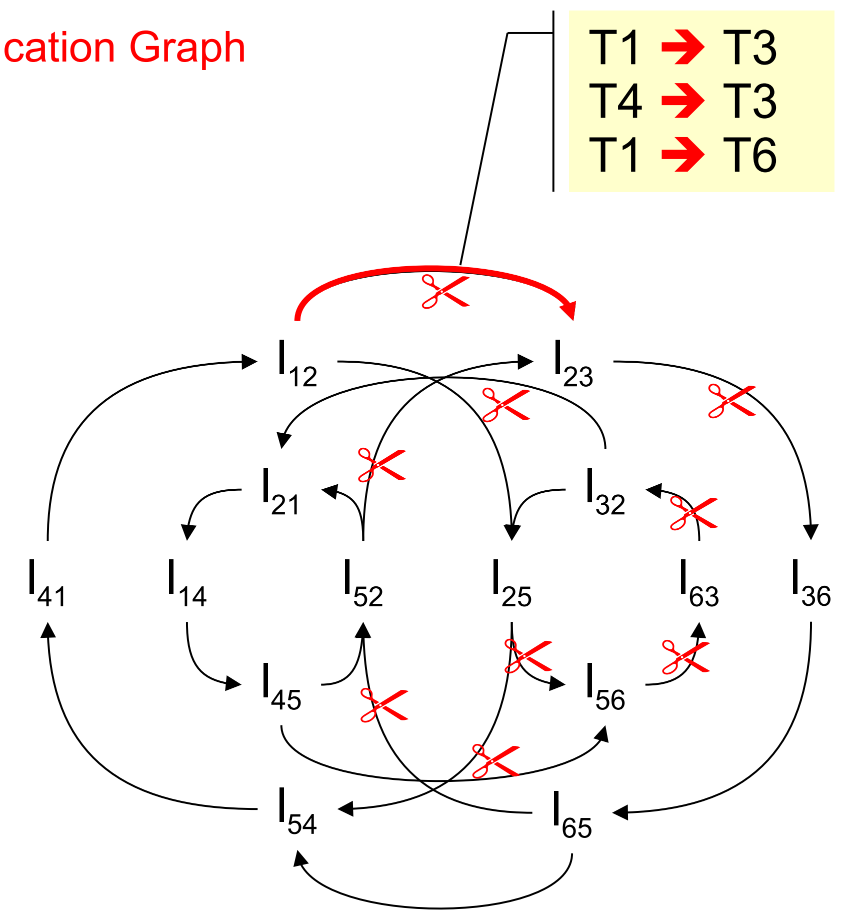


Communication Graph

T1 ➔ T3
T4 ➔ T3
T1 ➔ T6

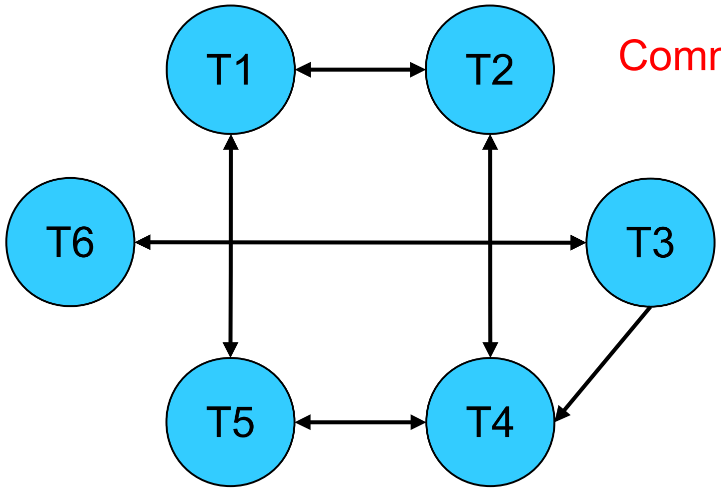
Topology Graph



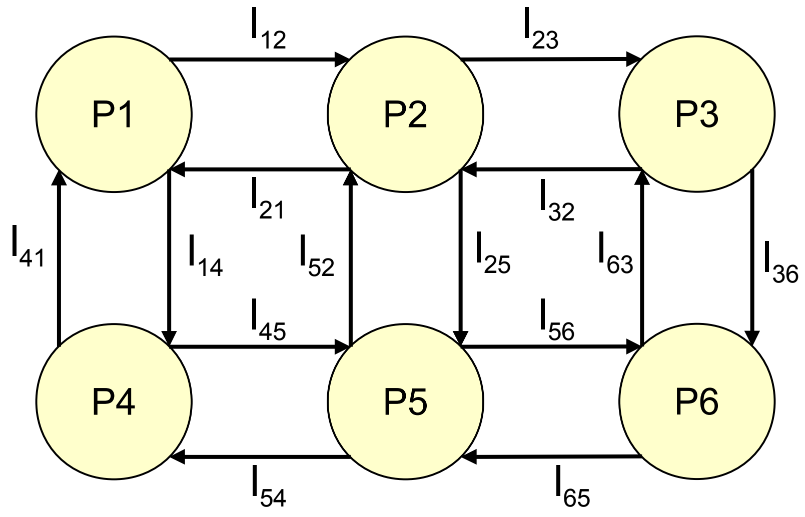


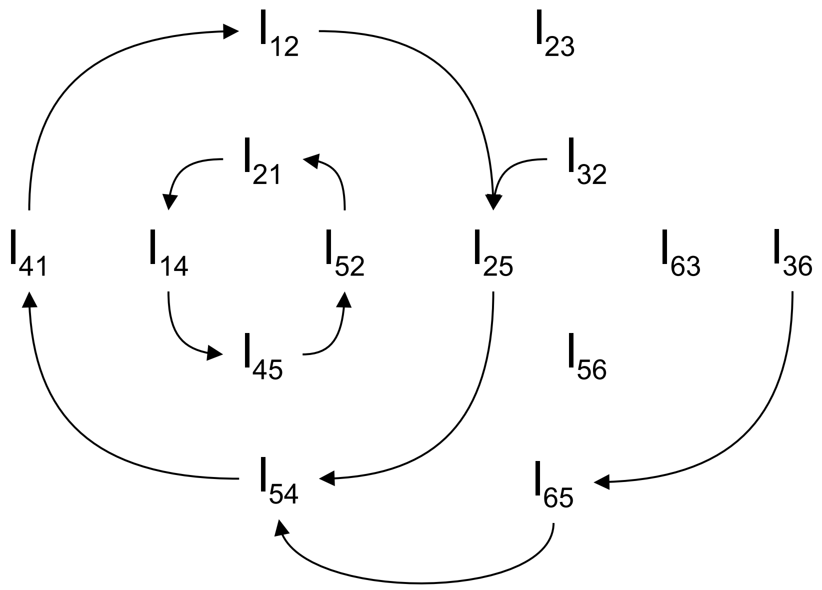
Channel Dependency Graph

APSRA by Example

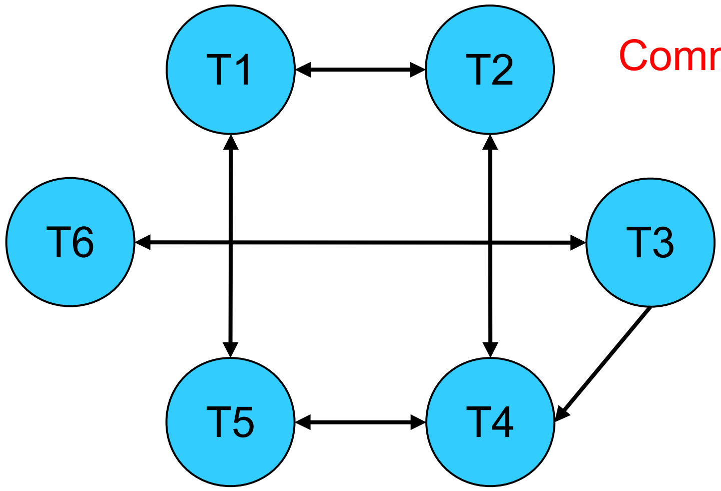

Communication Graph

Topology Graph



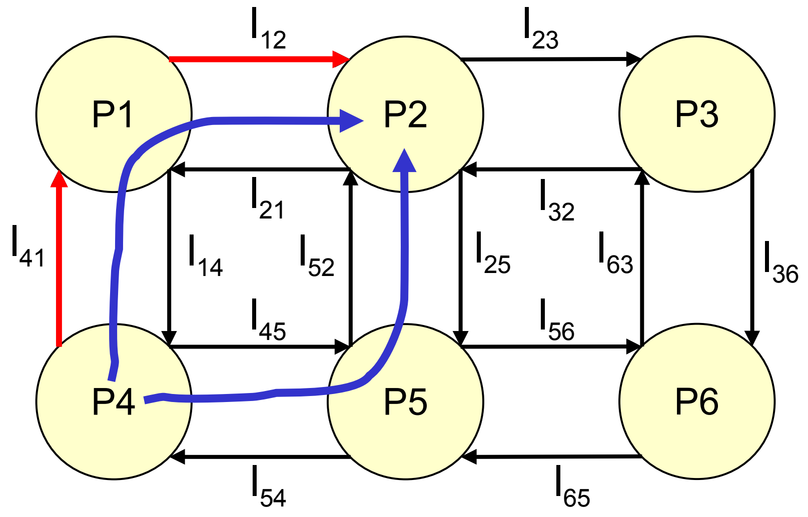

Application Specific
Channel Dependency Graph

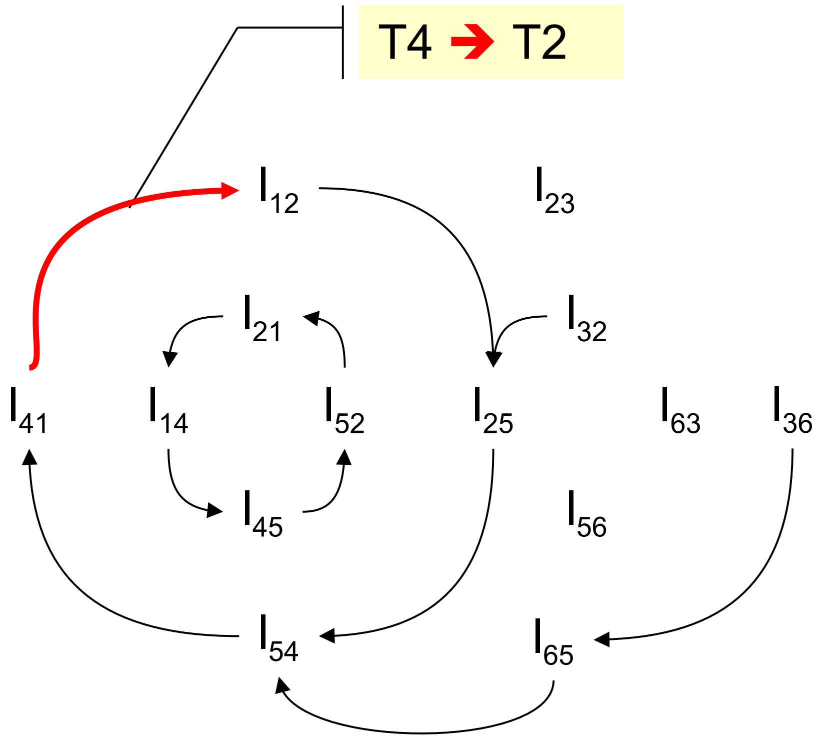
APSRA by Example



Communication Graph

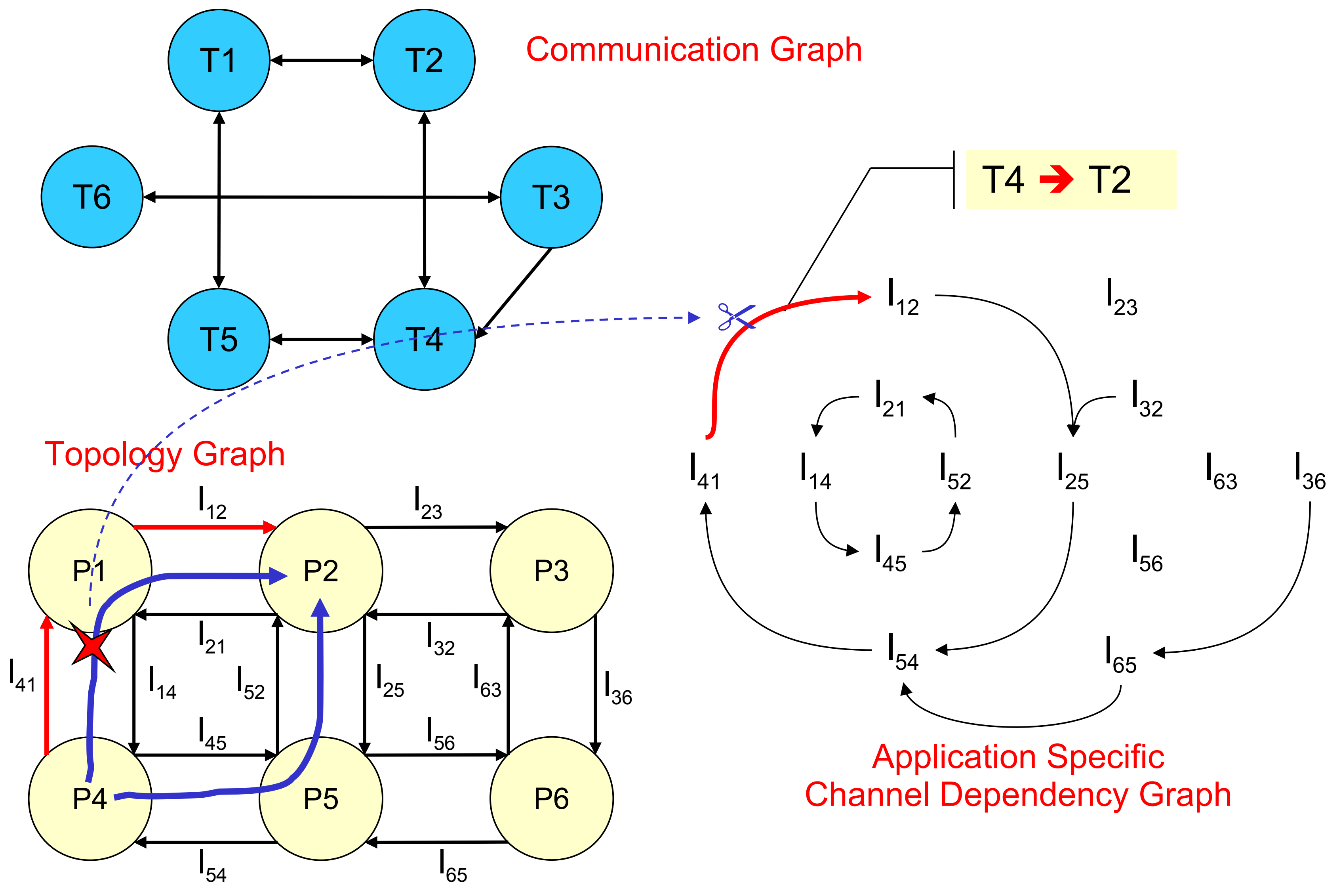
Topology Graph





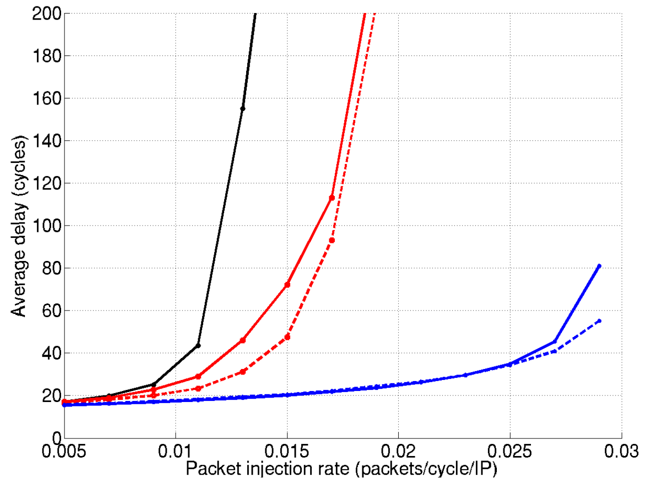
Application Specific
Channel Dependency Graph

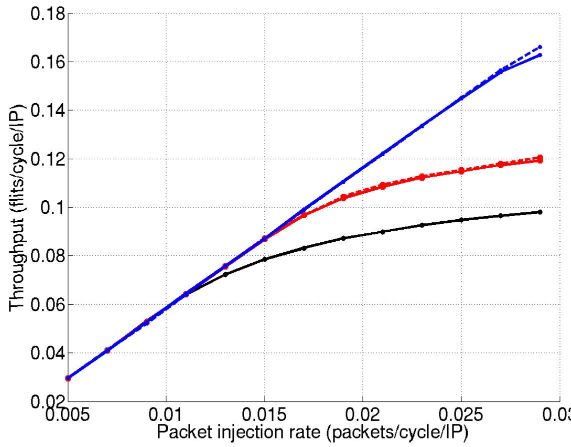
APSRA by Example



Performance Evaluation – Delay and Throughput

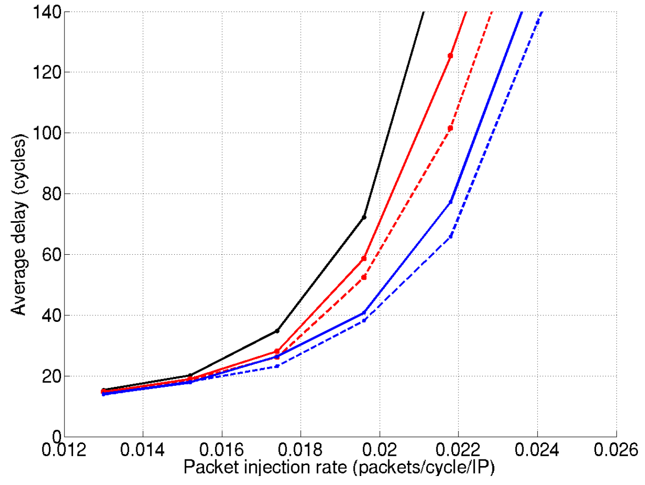
Transpose 1

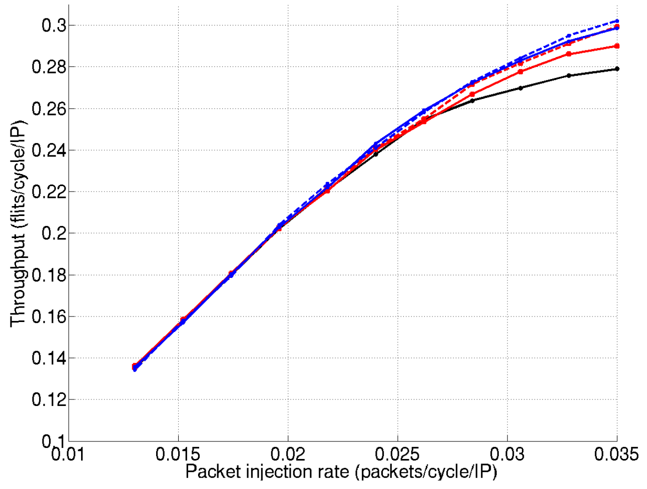


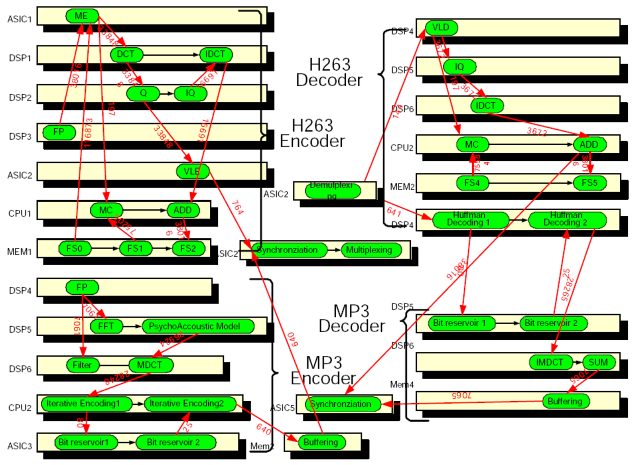


- ▬ XY
- ▬ Odd-Even (sel=random)
- ▬ APSRA (sel=random)
- ▪▪▪▪ Odd-Even (sel=buffer level)
- ▪▪▪▪ APSRA (sel=buffer level)

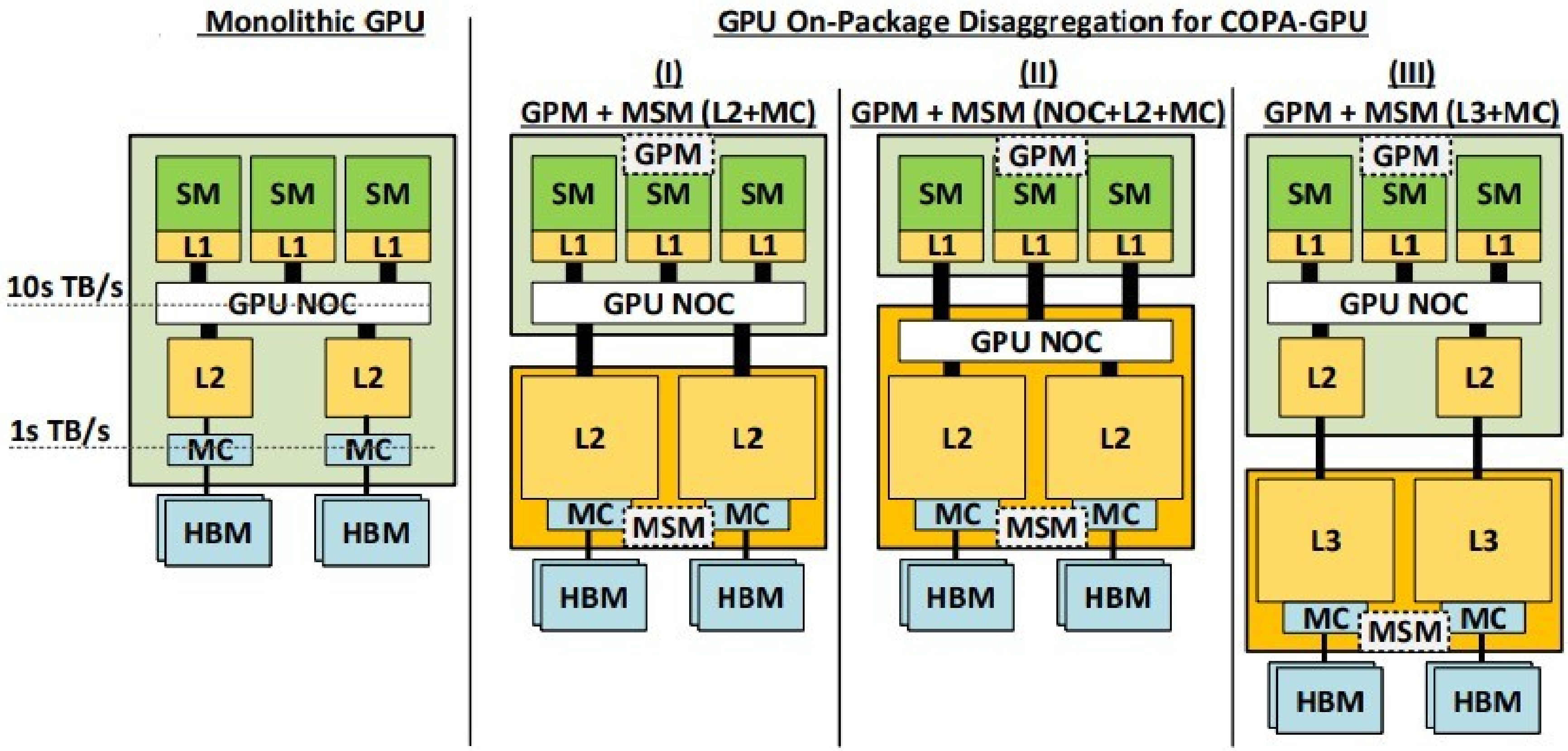
MMS



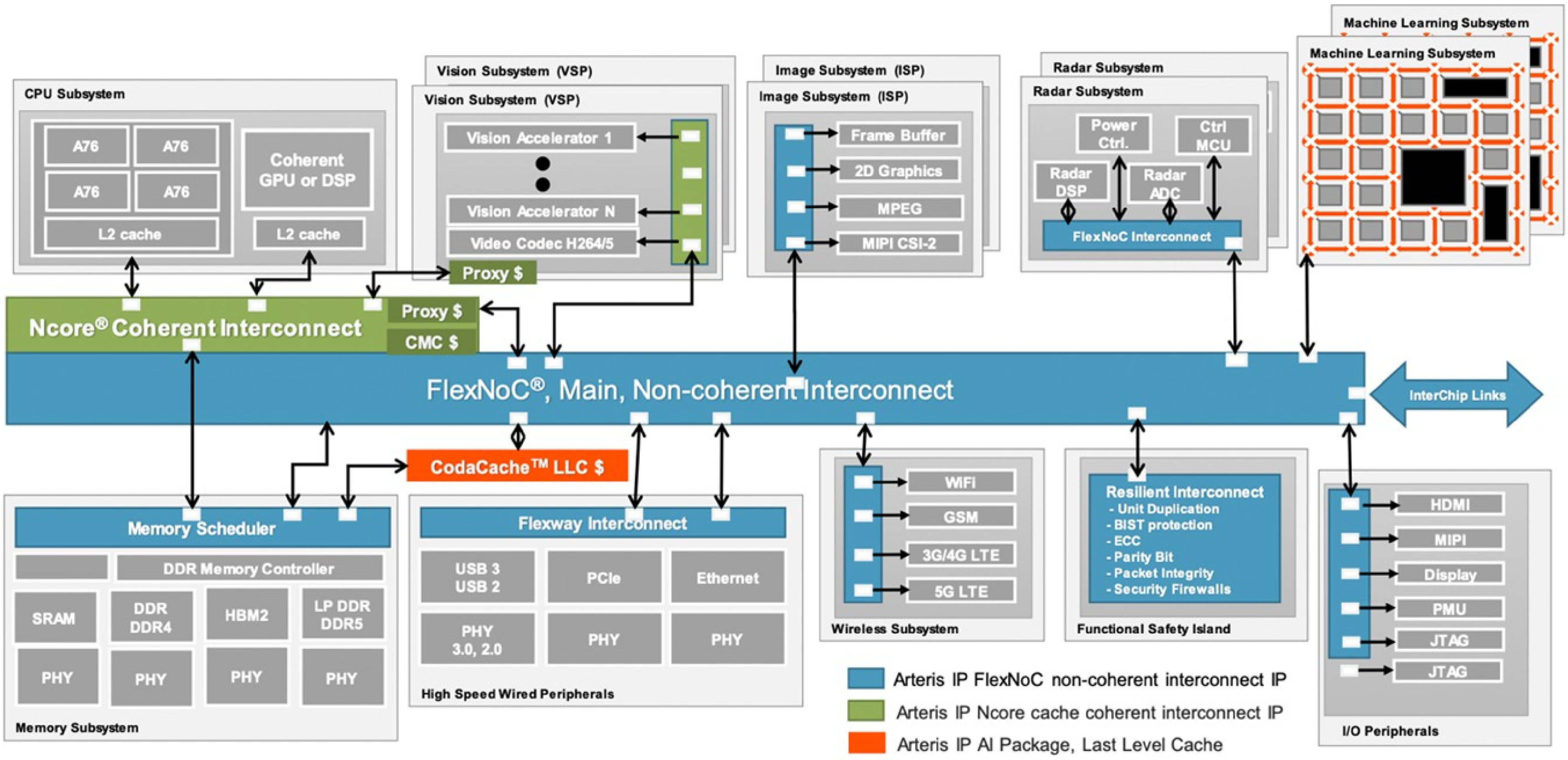




Multiple Multichip GPU Engine

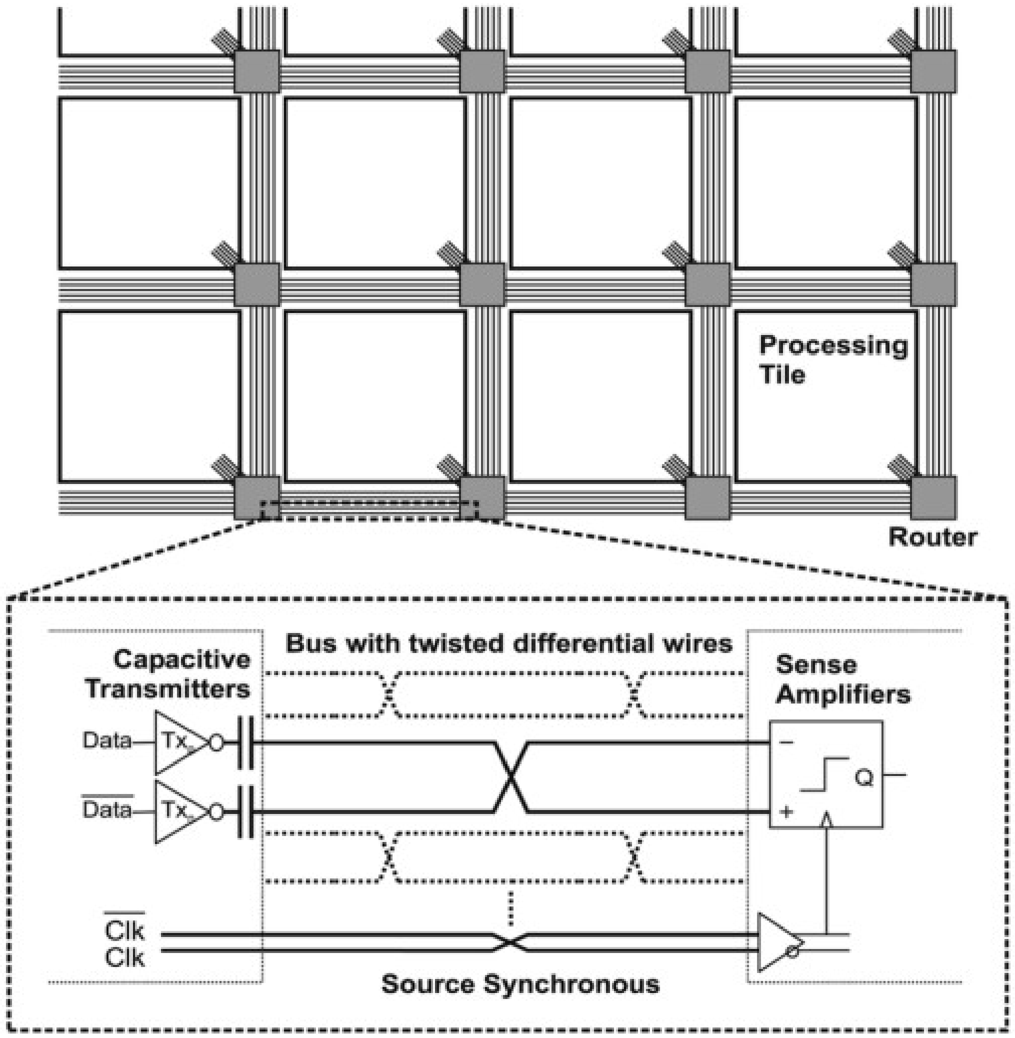


NoC based Design



Emerging Trends in NoC Architectures

Advanced NoC Design – Link



	Conventional full-swing	**Multi-V_{DD} low-swing**	**Capacitive low-swing**
Technology	1.2V, 6 metal, 90nm CMOS		
Interconnects	2mm, R_{wire}=400Ω		
	Shielded single ended C_{wire}=480fF	Twisted differential C_{wire}=560fF	
Supply	1.2V	V_{DDH} = 1.2V V_{DDL} = 1.08V	1.2V
Voltage swing	1.2V	120mV	120mV
Driver size	W_n=8μm W_p=20μm	W_p=20μm	W_n=1.6 μm W_p=4μm
Energy/trans Total Wire (theory) Tx overhead	420fJ 346fJ 74fJ	135fJ 8fJ 127fJ	105fJ 80fJ 25fJ
Static power	~9nW (leakage)	~10nW (leakage)	6μW
Data-rate 50% eye opening zero eye opening	fully shielded 5Gb/s 9Gb/s	5Gb/s 9Gb/s	9Gb/s 12Gb/s
Delay (50%) Transmitter *Nominal /Slow* *(T=25°C/T=100°C)* Interconnect	105ps/150ps 100ps	90ps/130ps 115ps	60ps/80ps 80ps

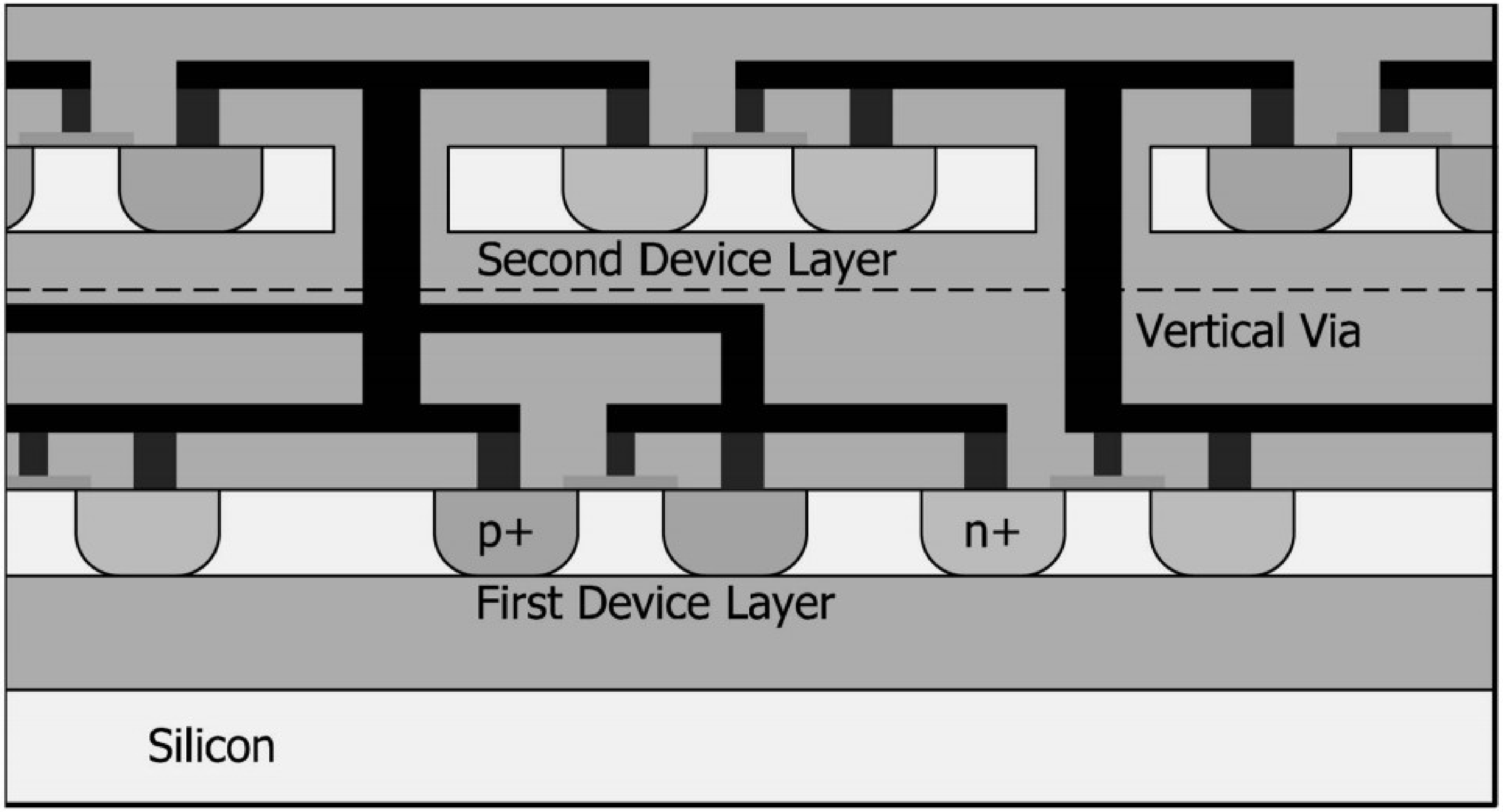
Outline

- Three-dimensional NoC
- RF interconnect
- Nanophotonic
- Wireless NoC

Outline

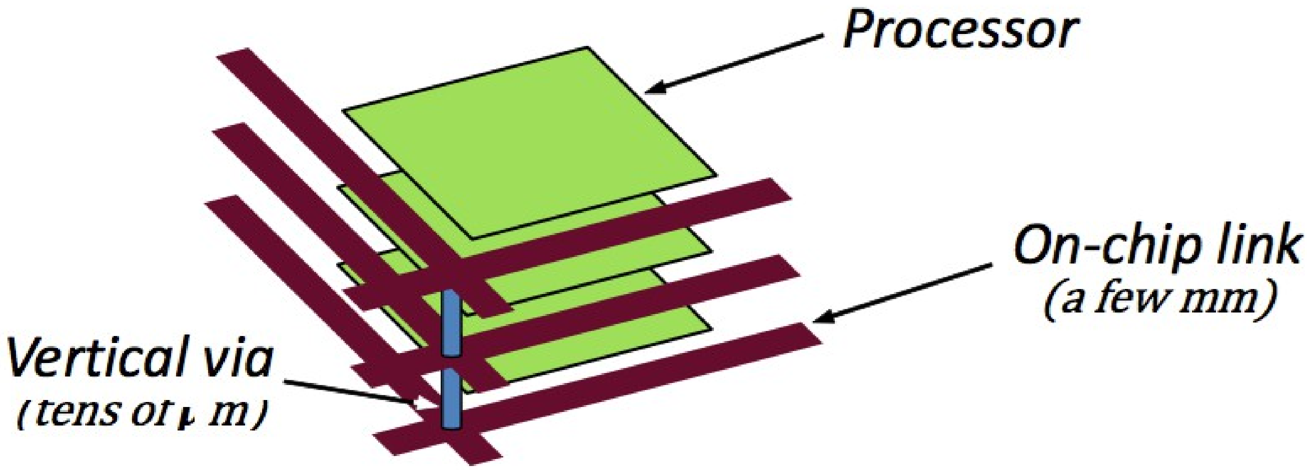
- Three-dimensional NoC
- RF interconnect
- Nanophotonic
- Wireless NoC

3D Integration



3D Integration

- Benefits
  - Higher packing density
  - Improved noise immunity
  - Overall superior performance



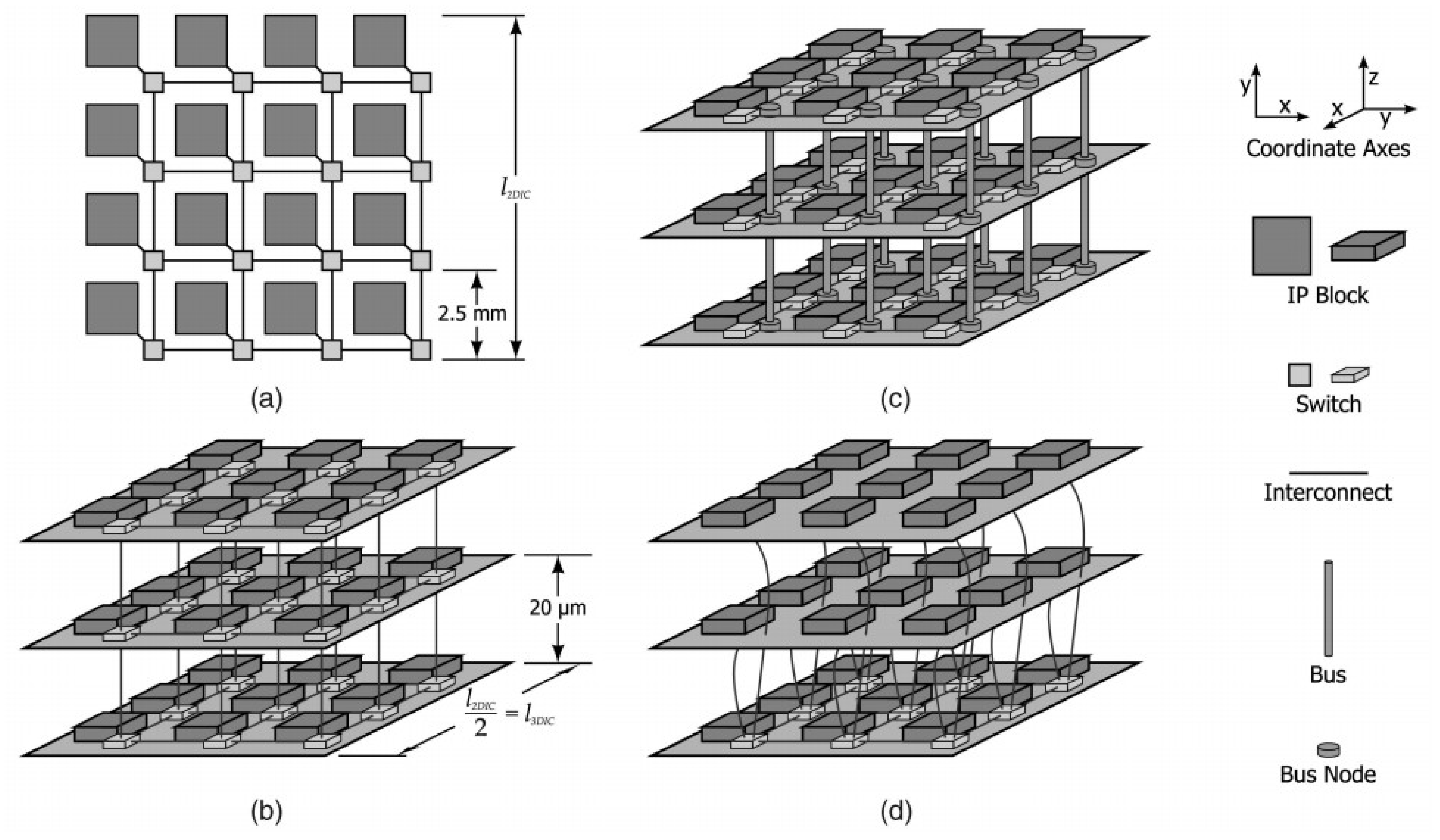
Vertical Interconnects

- Near-field coupling schemes
  - Virtually eliminate the need for a physical interconnection between layers
- Inductive coupling schemes
  - Longer transmission ranges than capacitive ones for similar data rates
  - N. Miura, *et al.* “A High-Speed Inductive-Coupling Link With Burst Transmission,” IEEE Journal of Solid-State Circuits, 44(3), 2009
- Capacitive coupling schemes
  - Area overhead approximately one order of magnitude lower than the inductive options
  - Q. Gu, *et al.* “Two 10Gb/s/pin Low-Power Interconnect Methods for 3D ICs,” Solid-State Circuits Conference, 2007
- Energy required per transmission substantially higher than state-of-the-art vias

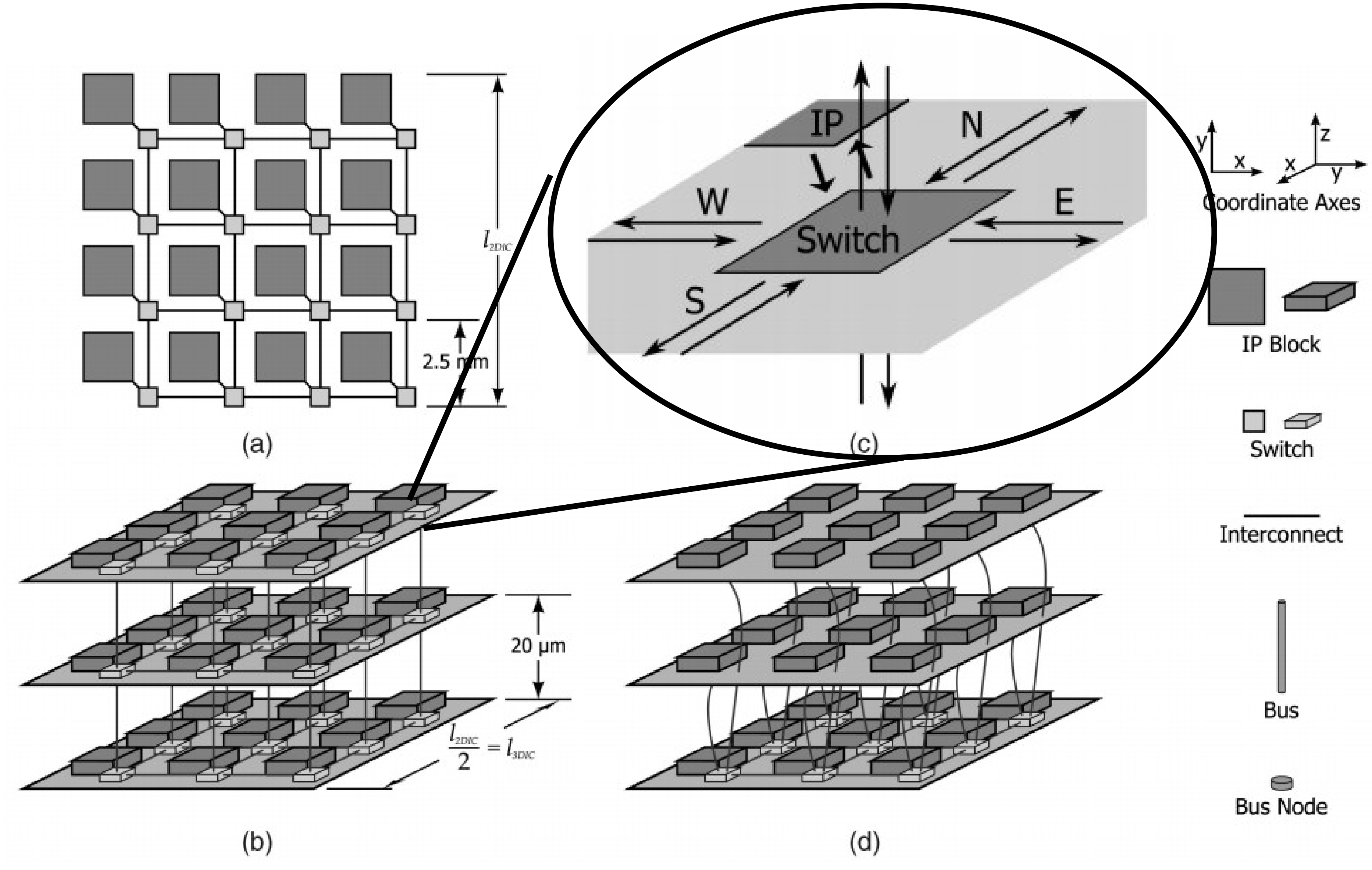
3D Integration Pros

- Significant reduction in terms of average propagation delay and energy per bit
  - Given by the short distance between layers of processors

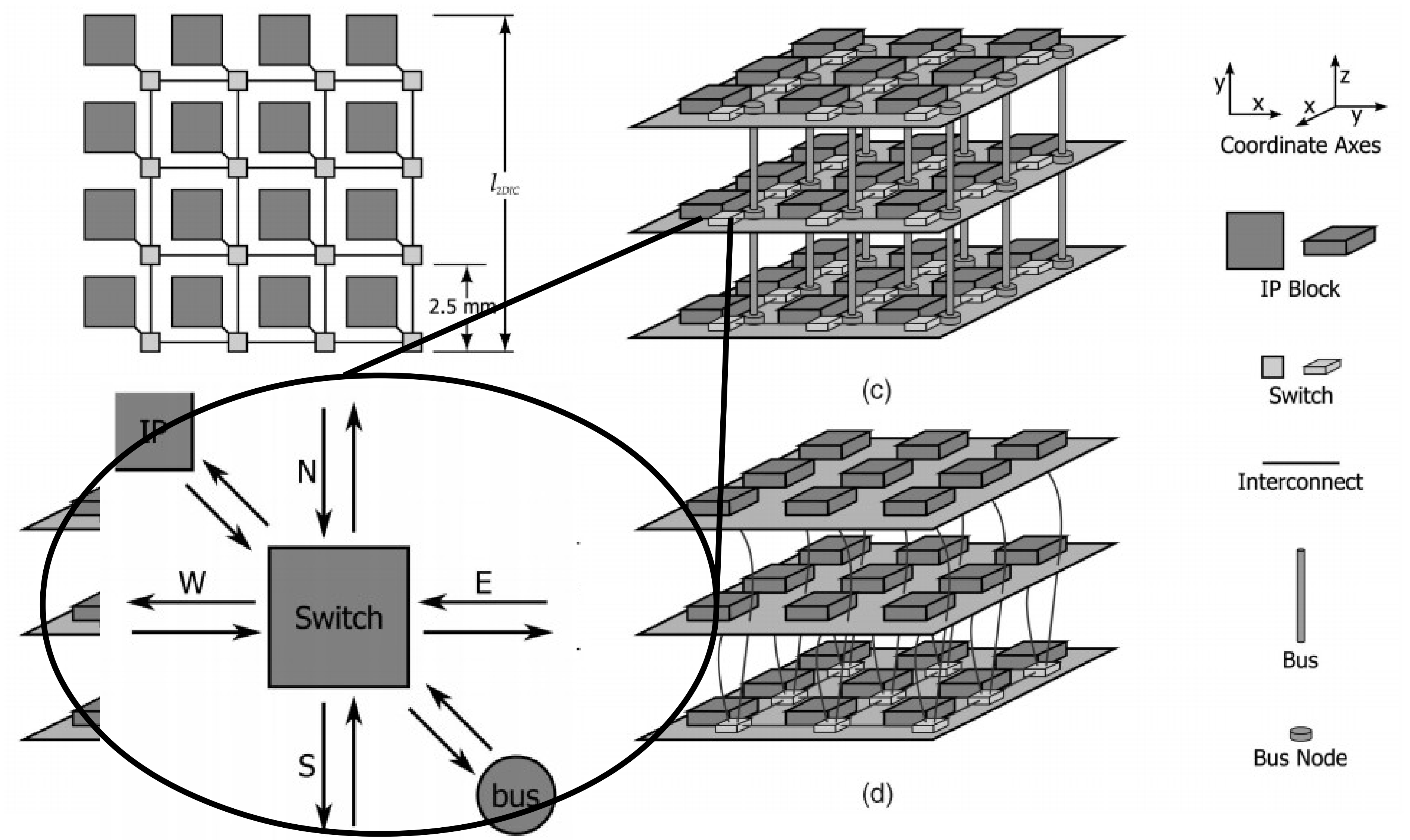
Mesh Based Architectures



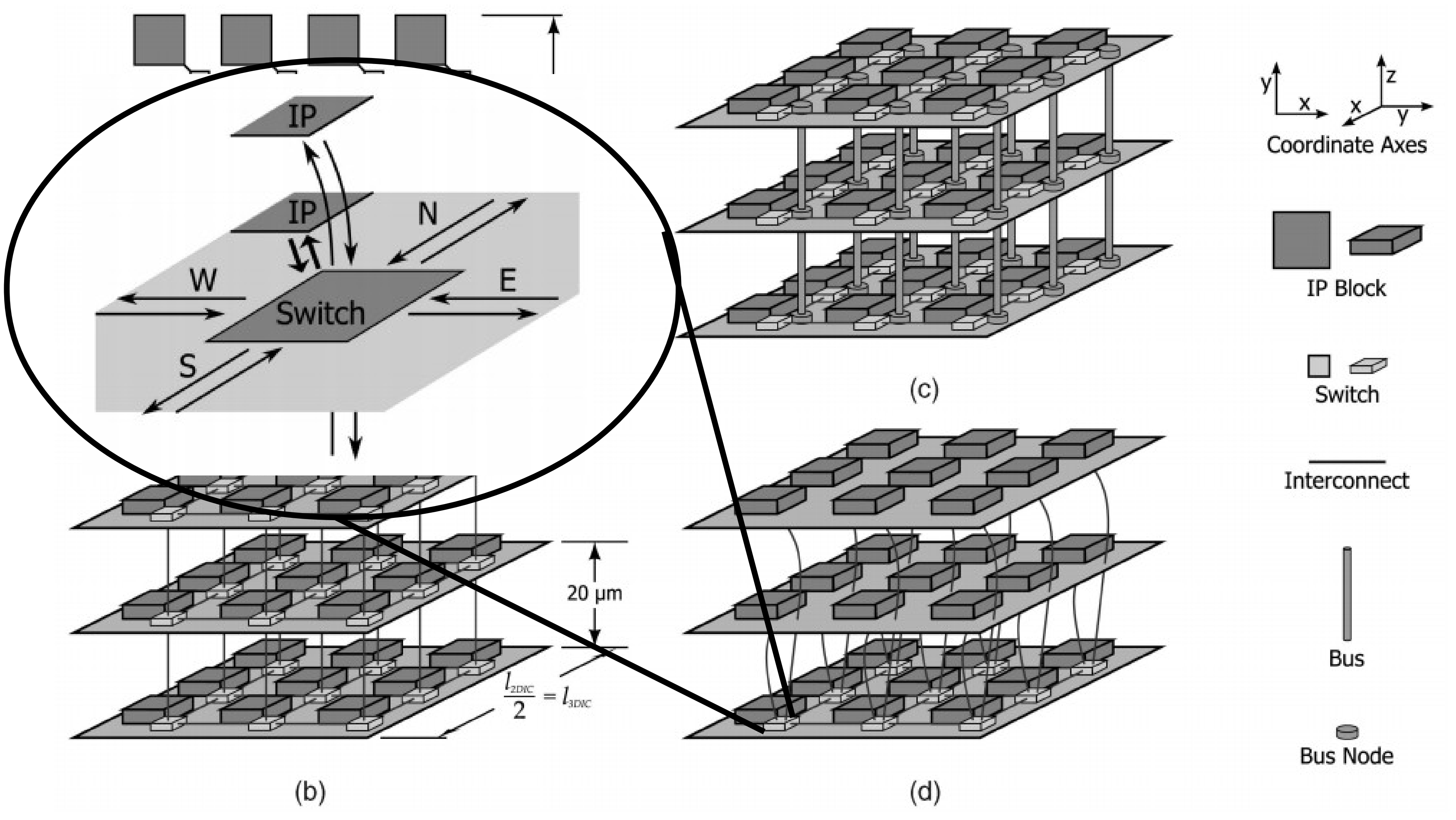
Mesh Based Architectures



Mesh Based Architectures



Mesh Based Architectures



3D Integration Pros

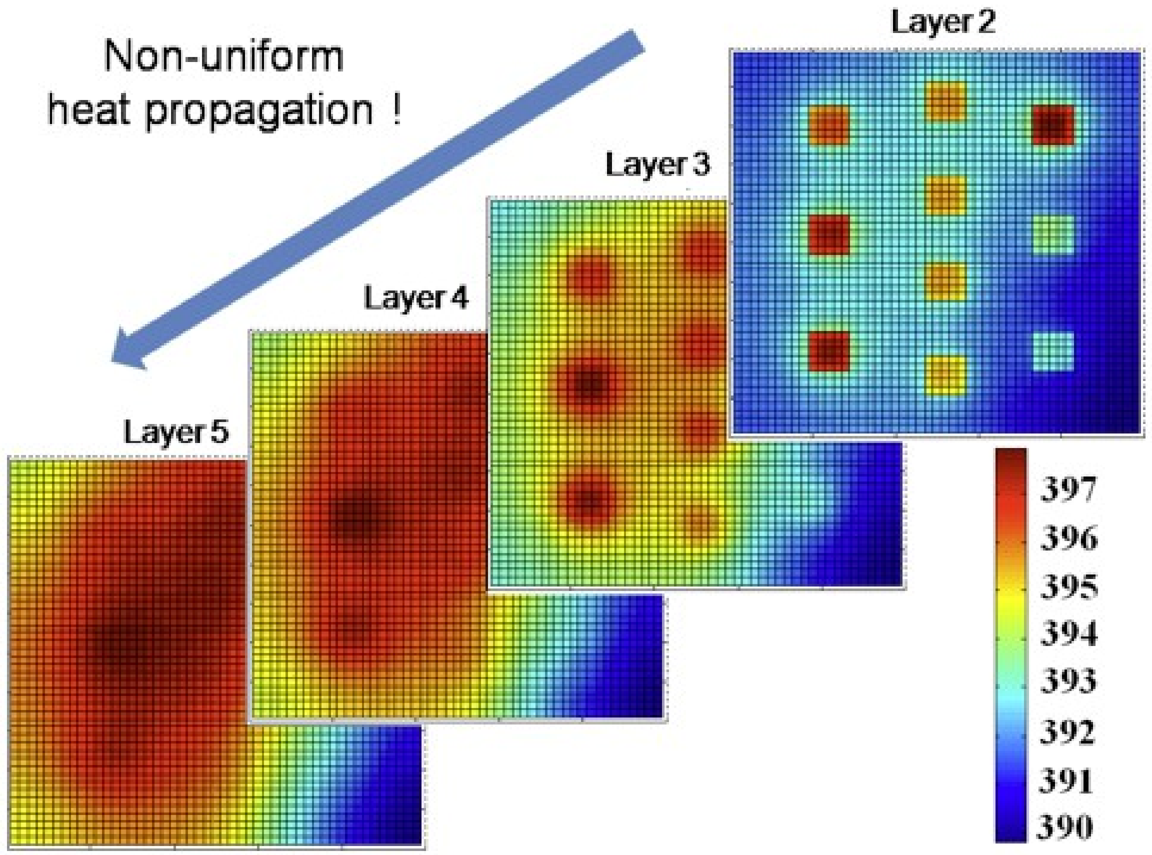
- Heterogeneous integration
  - Interface different technologies in hybrid approaches
    - Facilitating modularity by avoiding the integration of different technologies within the same layer
    - Circuit layers can be built with different processes, or even on different types of wafers
      - Components can be optimized to a much greater degree than if they were built together on a single wafer
      - Components with incompatible manufacturing could be combined in a single 3D IC

3D Integration Cons

- The super-position of active layers produces an increase in the heat density
  - V. S. Nandakumar and M. Marek-Sadowska, “A Low Energy Network-on-Chip Fabric for 3-D Multi-Core Architectures,” IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 2, no. 2, pp. 266–277, 2012
    - Reduces the power of TSV alleviating the heat density issues

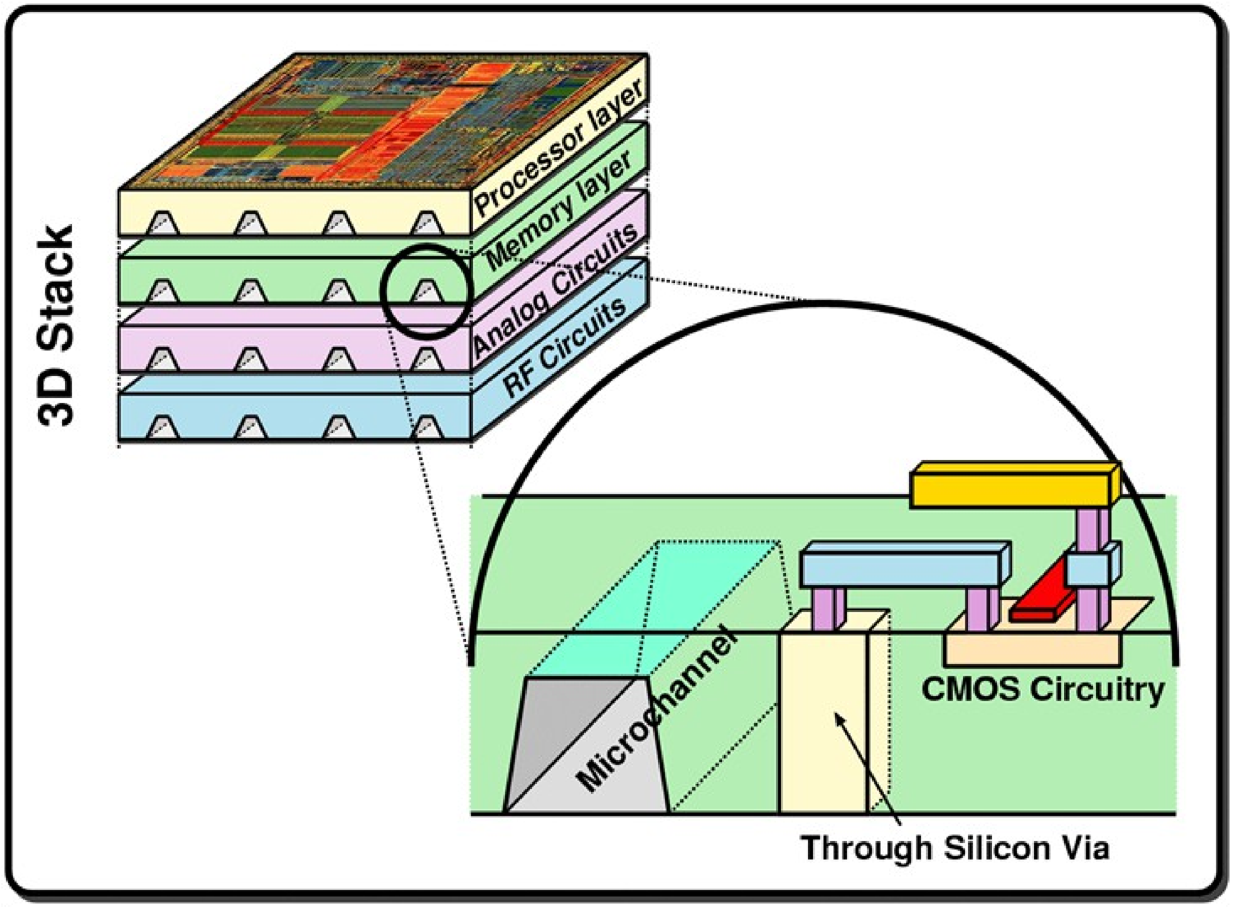
3D Integration Cons

- Thermal issues



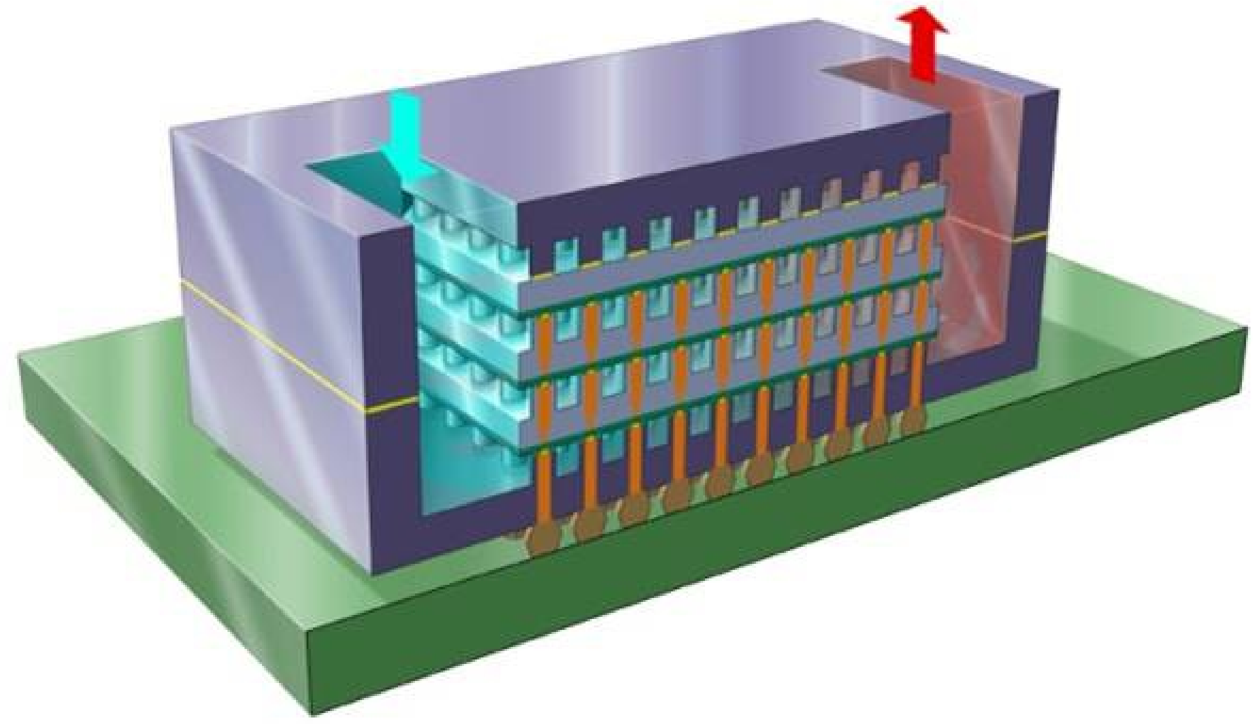
3D Integration Cons

- Liquid cooling



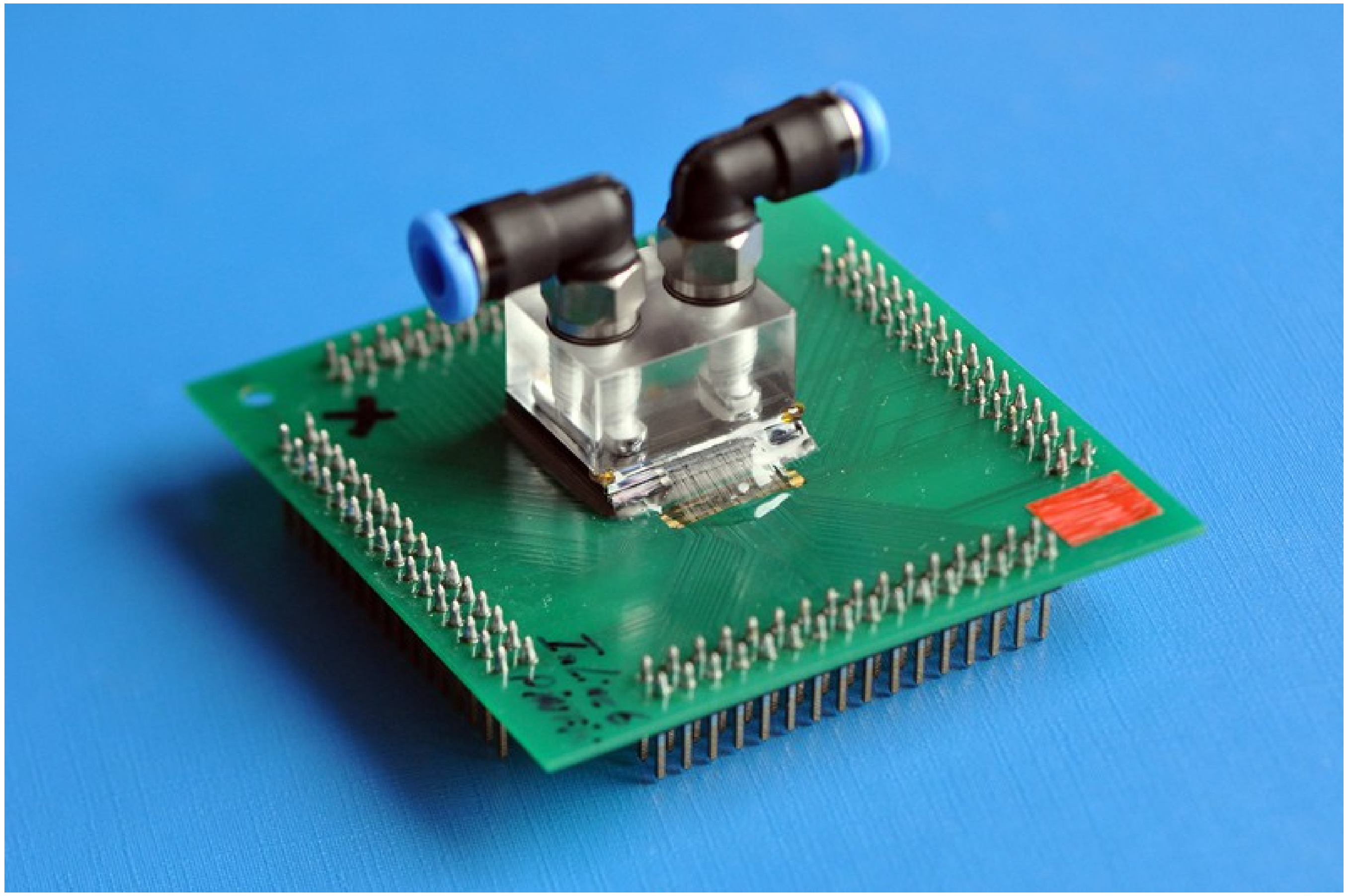
3D Integration Cons

- Liquid cooling



3D Integration Cons

- Liquid cooling



3D Integration Cons

- Alignment methodlogies are required for the precise positioning of the vias
- Wireless coupling schemes could eliminate this issue
  - At the cost of higher power consumption

Outline

- Three-dimensional NoC
- RF interconnect
- Nanophotonic
- Wireless NoC

RF Interconnects

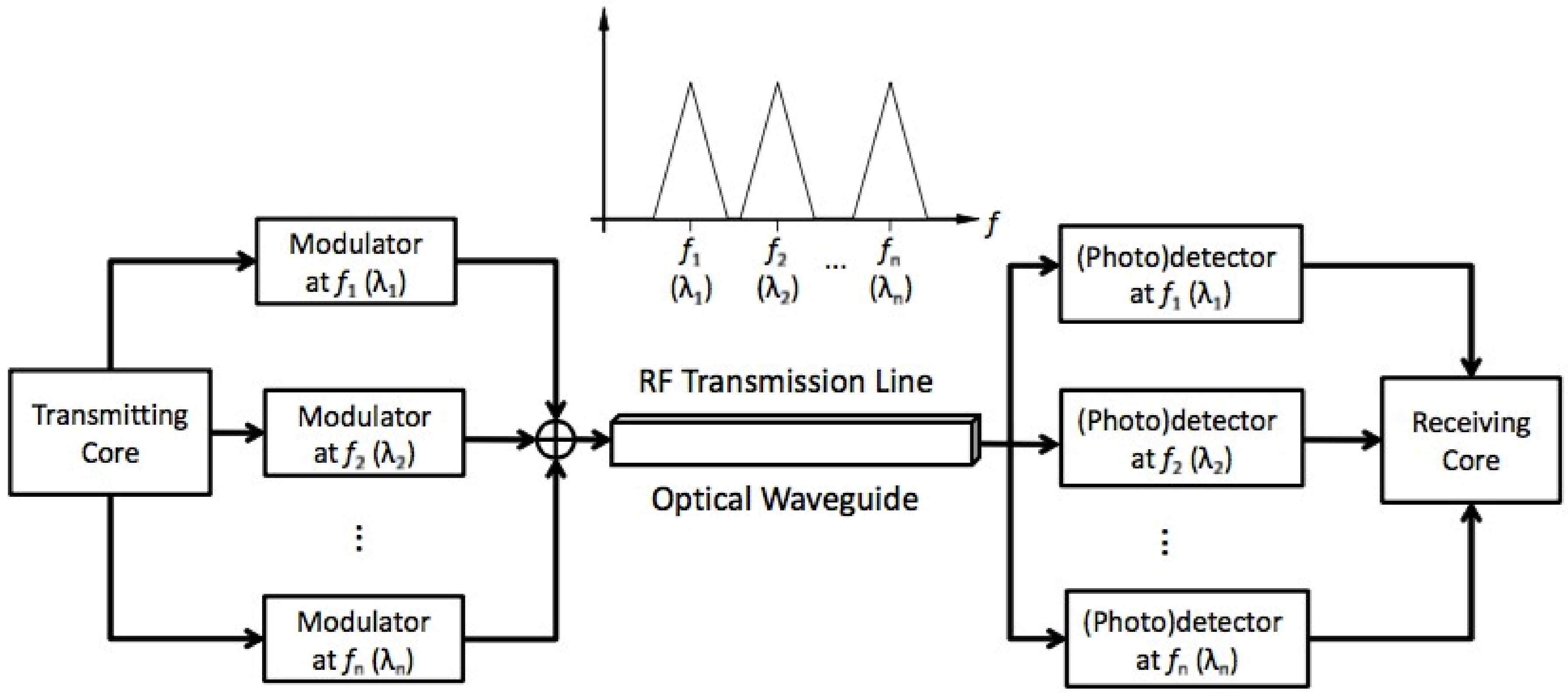
- Alternative to traditional voltage and current signaling through metallic wires
- Transmission of electromagnetic waves over microstrip transmission lines within the metal layers of the chip
  - E. Socher and M.-C. F. Chang, “Can RF Help CMOS Processors?” IEEE Commu- nications Magazine, vol. 45, no. 8, pp. 104–111, 2007
  - M.-C. F. Chang, V. Roychowdhury, L. Zhang, H. Shin, and Y. Qian, “RF/wireless in- terconnect for inter- and intra-chip communications,” Proceedings of the IEEE, vol. 89, no. 4, pp. 456–466, 2001

RF Interconnects

- The original baseband signals are modulated using carrier waves at frequencies up to several GHz and then guided through the transmission lines
- Signals propagate at the speed of light instead of at the charging and discharging speed of RC wires
  - Need to be taken back at the baseband frequency and demodulated at the receiving end

RF Interconnects

- Transmission lines can have multiple inlets and outlets, allowing to design multi-receiver schemes
- Shared transmission lines enable the use of multiplexing techniques



f
f_1 (λ_1)
f_2 (λ_2)
...
f_n (λ_n)
Transmitting Core
Modulator at f_1 (λ_1)
Modulator at f_2 (λ_2)
⋮
Modulator at f_n (λ_n)
RF Transmission Line
Optical Waveguide
(Photo)detector at f_1 (λ_1)
(Photo)detector at f_2 (λ_2)
⋮
(Photo)detector at f_n (λ_n)
Receiving Core

RF Interconnects Pros

- Speed-of-light propagation
  - In long-range links, the improvement in terms of propagation time is very large with respect to the delay introduced by the modulation process and, therefore, the communication latency can be effectively reduced and be made virtually independent of the link length

RF Interconnects Pros

- Possibility of transmitting several frequency-orthogonal signals through the same transmission line
  - Each core can be assigned a set of channels, so that several cores are interconnected using the same transmission line, thereby reducing the number of wires

RF Interconnects Cons

- Circuital implementation of frequency multiplexing transceivers produces both an area and power overhead that must be reduced before the whole design can be scaled

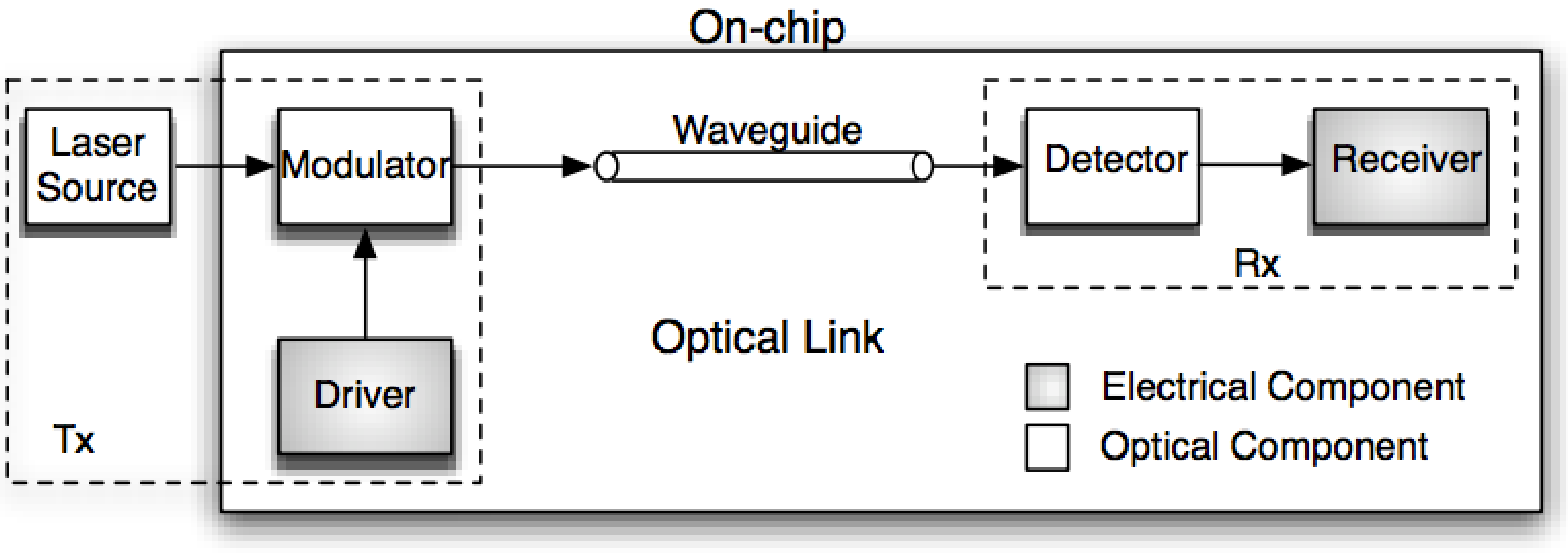
RF Interconnects Cons

- The physical topology must be carefully designed as impedance mismatch reflections at the terminations of the transmission line may generate interferences
  - This strongly limits the number of nodes connected through the same transmission line
  - Forces the use of amplifiers that bridge different transmission line segments

Outline

- Three-dimensional NoC
- RF interconnect
- Nanophotonic
- Wireless NoC

Nanophotonics

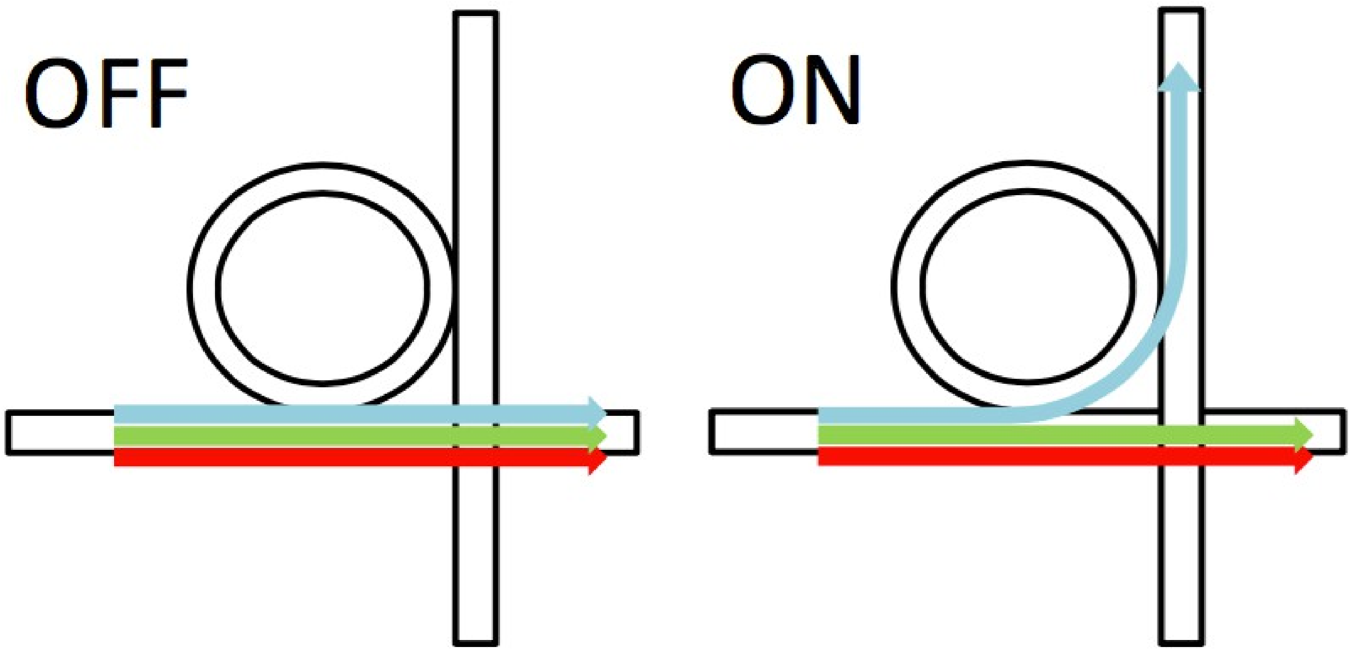


Nanophotonics

- Nanophotonics support the transmission of signals at different frequencies
  - Better referred here to as wavelengths
  - At a much higher density than in RF interconnects

Nanophotonics

- Microring resonators



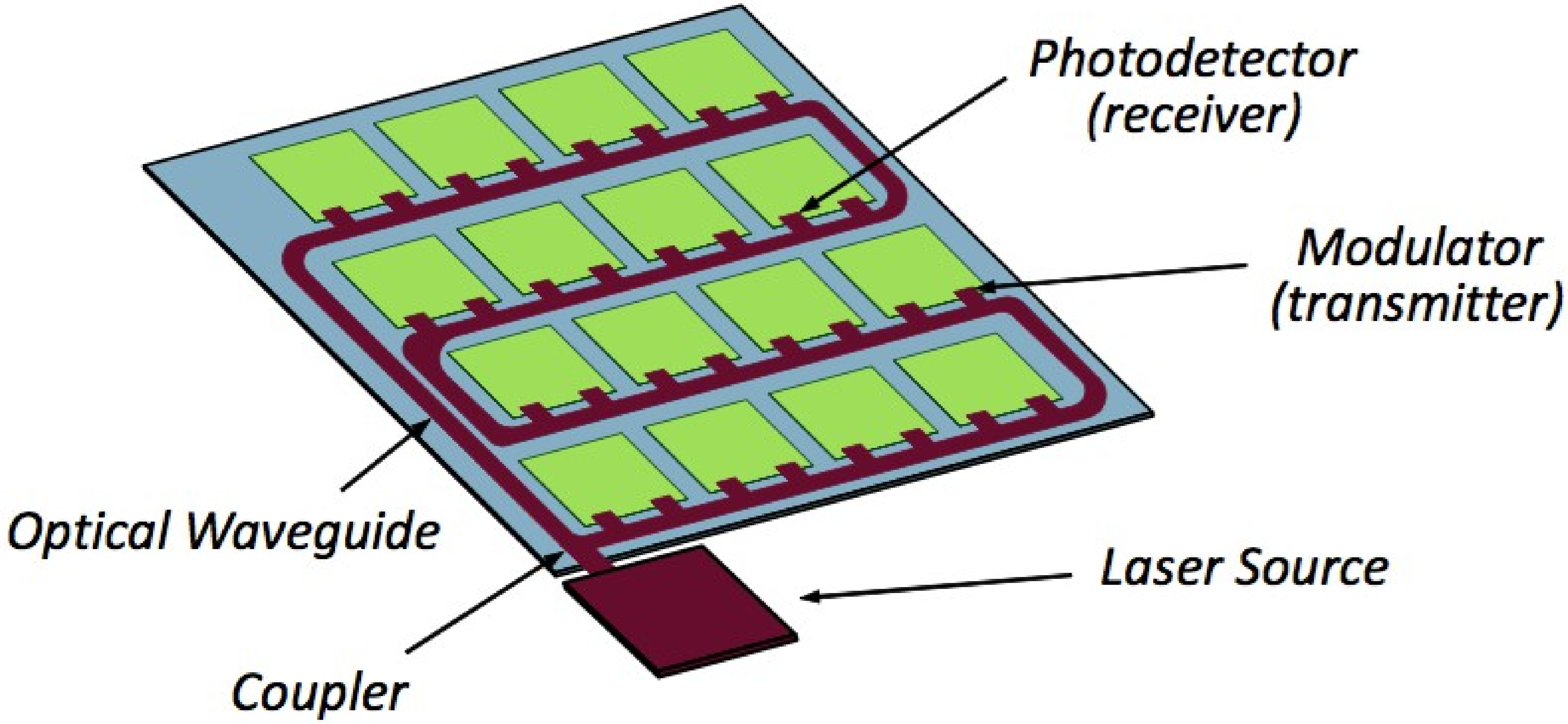
Nanophotonics Pros

- Particular and extreme case of RF interconnect
  - Optical signals are basically electromagnetic waves at much higher frequency than RF signals
- Share some of the advantages of RF
  - Relative flexibility provided by multiplexing
  - Low latency potential given by the speed-of-light propagation
  - Bandwidth density
    - Much higher than in other emerging alternatives due to the very small wavelengths involved in the communication
  - Energy efficiency orders of magnitude better than that of RC wires

Nanophotonics Cons

- Impracticality of light buffering and header processing prevents the creation of optical routers for on-chip communication

Photonic Network-on-Chip



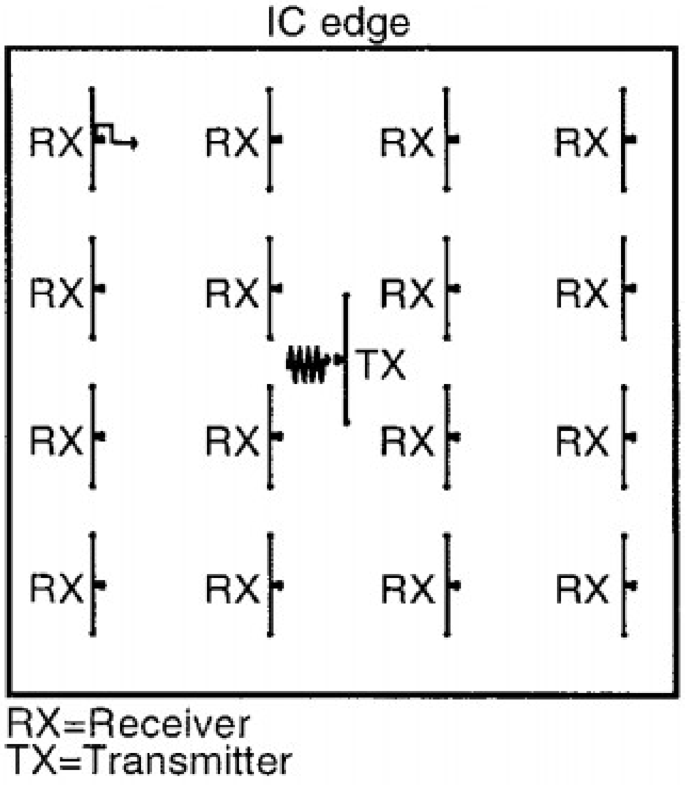
Outline

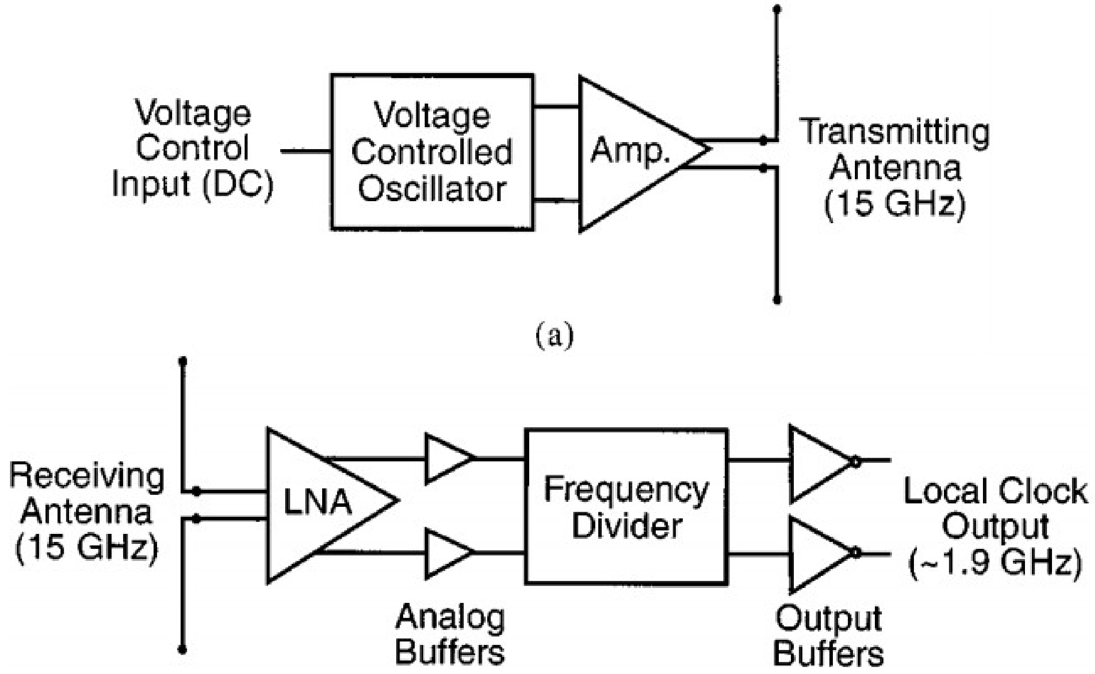
- Three-dimensional NoC
- RF interconnect
- Nanophotonic
- Wireless NoC

Wireless On-Chip Communication

- Relatively recent field
- B. A. Floyd, *et al.*, “Intra-chip wireless interconnect for clock distribution implemented with integrated antennas, receivers, and transmitters,” IEEE Journal of Solid-State Circuits, 37(5), 2002
  - First pioneering work appeared in 2002
  - On-chip antenna to wirelessly distribute the clock signal across a single chip
  - Validated with a test chip integrating both the transmitter and the receiver with their respective zig-zag antennas at 15 GHz

Wireless On-Chip Communication



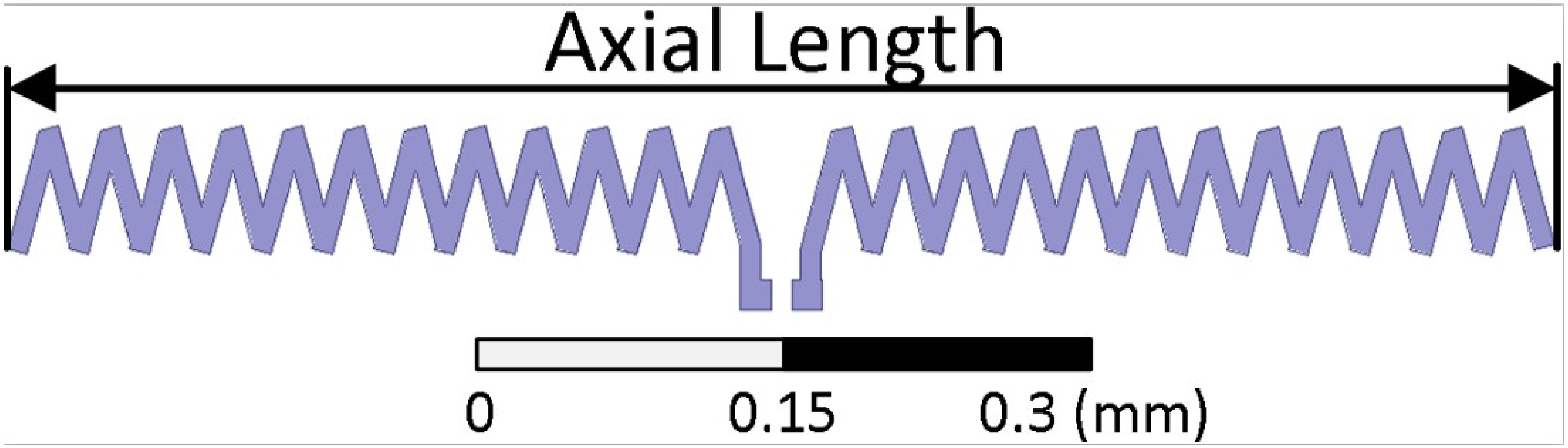

Voltage Control Input (DC)
Voltage Controlled Oscillator
Amp.
Transmitting Antenna (15 GHz)
(a)
Receiving Antenna (15 GHz)
LNA
Analog Buffers
Frequency Divider
Output Buffers
Local Clock Output (~1.9 GHz)

Wireless On-Chip Communication

- The work paved the way for the development of the WiNoC paradigm
- The approach is possible by virtue of the steady technology scaling trends
  - Millimeter-wave (mmWave) frequencies
  - Scaled the antennas and passive elements of the circuits that drive the antennas down to sizes much smaller than the silicon chips where they are integrated

mmWave Antenna

- On-chip zig-zag antennas
  - Quarter-wave axial length of around 0.38 mm
  - Width on the order of 0.01 mm
  - Operating at 60-GHz



Wireless *vs.* RF Communication

- Share the basic working principles
  - Different propagation medium
    - Waveguide *vs.* chip package
  - A transmitting antenna modulates information and radiates the resulting RF signals at a given frequency
  - These waves propagate within the chip package following different propagation mechanisms
  - May be received by any other antenna tuned to the same frequency and within the range of the transmitter

Wireless *vs.* RF Communication

- All antennas tuned to the same frequency channel share the medium similarly to in a bus or a transmission line

Wireless Communication Pros

- Latency advantage with respect to RC signaling
  - Wave propagation occurs at the speed of ligh
- Improved simplicity, flexibility, and reconfigurability potential
  - No path infrastructure is required between node
  - Can modify the logical topology or other transmission parameters without the need of any physical modification

Wireless Communication Cons

- Energy is radiated through space rather than guided in through planar 2D structure
  - Wireless communication is intrinsically less efficient
- Lack of an isolated medium for propagation
  - All antennas tuned to the same frequency need to either compete or be scheduled to access to the medium
  - Negative impact on bandwidth

Comparative Summary

	Baseline	3D NoC	Nanophotonics	RF-I	Wireless
Wiring	RC Wires	Vertical Vias	Waveguides	Transmission Lines	None
Principle	Wire charge	Wire charge	Optical signals	Guided RF signals	Radiated RF signals
Propagation Speed	Large	Short	Speed of light	Speed of light	Speed of light
Technological Availability	Now	Now	Mid term	Short term	Short term
Integration Complexity	Low	Low	Very high	High	High
Bandwidth Density	Good	Better	Best	Better	Modest
Intrinsic Efficiency	Good	Better	Best	Better	Modest
Architectural Flexibility	Average	Average	Low	Low	High
Issues	Wire delay	Thermal Effects	Laser power, buffering	Signal reflections	Multiple access